(12)

PROCEEDINGS:

IMAGE UNDERSTANDING WORKSHOP

SEPTEMBER 1982

Sponsored by:
Information Processing Techniques Office
Defense Advanced Research Projects Agency

Hosted by;
Stanford University
Palo Alto, California

DEPTH FROM STEREO

Science Applications, Inc.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SAI-83-892-WA	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) IMAGE UNDERSTANDING Proceedings of a Workshop, September, 1982		5. TYPE OF REPORT & PERIOD COVERED SEMI ANNUAL TECHNICAL May, 1981 -September, 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) LEE S. BAUMANN (Ed.)		8. CONTRACT OR GRANT NUMBER(s) MDA903-82-C-0160
9. PERFORMING ORGANIZATION NAME AND ADDRESS SCIENCE APPLICATIONS, INC. 1710 Goodridge Drive, 10th Floor McLean, Virginia 22102		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA ORDER No. 3456
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE September, 1982
		13. NUMBER OF PAGES 351
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing; Image Understanding; Scene Analysis; Edge Detection; Image Segmentation; CCDArrays; CCD Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the technical papers and outlines of semi-annual progress reports presented by the research activities in Image Understanding, sponsored by the Information Processing Techniques Office; Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 15-16 September 1982 in Palo Alto, California.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

IMAGE UNDERSTANDING

Proceedings of a Workshop
Held at
Palo Alto, California
September 15-16, 1982

Sponsored by the
Defense Advanced Research Projects Agency

Science Applications, Inc
Repoprt Number SAI-83-892-WA
Lee S. Baumann
Workshop Organizer and
Proceedings Editor

This report was supported by
The Defense Advanced Research
Projects Agency under DARPA
Order No. 3456, Contract No. MDA903-82-C-0160
Monitored by the
Defense Supply Service, Washington, D.C.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

Foreword

The Thirteenth Image Understanding (IU) Workshop sponsored by the Defense Advanced Research Projects Agency (DARPA), Information Processing Techniques Office, was held at Stanford University, Palo Alto, California, on 15-16 September 1982. The Intelligent Systems Program Manager, Cdr. Ronald B. Ohlander, USN, welcomed the more than one hundred government, academic and industrial participants to the workshop. He noted that it was particularly fortuitous that this meeting, occuring some seventeen months since the last DARPA sponsored workshop in Washington, D.C. in April 1981, should take place in the Palo Alto, California area. The reason for this was because one of the principal participants since the beginning of the program, Stanford University, was the host and two of the premier Artificial Intelligence Laboratories in the country, Stanford University and SRI, International, are located in the area and were available for visits by workshop participants. In addition, the cartographic testbed which has been established at SRI International has progressed at a very acceptable rate, and participants received a more detailed description, as well as demonstrations, by Dr. Andy Hanson of SRI on the afternoon of the second day of the workshop. Cdr. Ohlander also welcomed the participation of the Defense Mapping Agency (DMA) at the workshop and noted that their presentation at the afternoon session of the first day presented a view of DoD technology requirements that should have significant impact on future research directions.

Section I of these proceedings contains outlines of the program progress reports as presented in Session I by the seven principal investigators involved in the DARPA research program in Image Understanding. Session II of the workshop consisted of the DMA discussion entitled, "DMA's Long Range Plan for Technology Transfer and Its Relationship to the DARPA Community." This presentation, by Mr. Henry Cook and Major Dave Nelson of DMA, was followed by a general discussion. No report of this session is contained in these proceedings. Sessions III and IV consisted of eighteen technical papers presented during the afternoon of day one and the morning session of day two. The papers from these two sessions are reproduced in Section II of these proceedings. Session V of the workshop on the afternoon of day two was devoted to the visits to the Stanford University and SRI, International Artificial Intelligence Laboratories as described above. Since time did not permit the presentation of all technical papers prepared by the researchers in the DARPA Image Understanding Program, those papers not presented are reproduced in these proceedings in Section III. This procedure was adopted in order that a complete record be available to research and operational personnel requiring information on the subjects covered.

This workshop was hosted by Dr. Thomas O. Binford, Research Associate at the Artificial Intelligence Laboratory at Stanford University. The workshop organizer wishes to express the appreciation of all to Dr. Binford for his invaluable assistance in providing the necessary arrangements and facilities. Gratitude is also due to Ms. Marianne Siroker of the Stanford University staff who attended to all of the myriad of details necessary to arrange for and conduct the workshop. The success of the workshop is largely due to Ms. Siroker's efforts. Typing, support, including mailings, collection and arrangement of the conference proceedings, and assistance for the workshop in general, was provided by Miss Karen Long, Ms. Charlotte Dettor, and Ms. Lau[illegible] Weiss of the Science Applications, Inc. staff.

The materials for the cover of this document were submitted by Dr. Harlyn Baker of Stanford University. Dr. Baker notes that the urban stereo imagery is a synthetic pair generated by Control Data Corporation (for demonstrating their graphics capabilities some years ago), and shows a block-type depiction of roads and buildings of various heights. The depth map, states Baker, was produced by an edge and intensity based stereo matching process described in his paper on this subject in these proceedings. Dr. Baker believes that, in the future, depth maps of the sort depicted here will be used within the ACRONYM modelling and reasoning system for recognition of 3-dimensional structures. Both pairs of figures (the images and the depth map) are set up for stereoscopic viewing with cross-eyed fusion (left eye see right image of pair, right eye sees left image). The artwork and lay-out for the proceedings cover was created by Mr. Tom Dickerson of SAI.

Lee S. Baumann
Science Applications, Inc.
Workshop Organizer

DEFENSE TECHNICAL INFORMATION CENTER

ACCESSION NUMBERS FOR PREVIOUS

I.U. WORKSHOPS

ISSUE	DATE		NUMBER
APRIL	1977	ADA	052900
OCTOBER	1977	ADA	052901
MAY	1978	ADA	052902
NOVEMBER	1978	ADA	064765
APRIL	1979	ADA	069515
NOVEMBER	1979	ADA	077568
APRIL	1980	ADA	084764
APRIL	1981	ADA	098261

AUTHOR INDEX

NAME	PAGE
Baker, H. H.	215
Ballard, D. H.	79
Barnard, S. T.	193, 286
Binford, T. O.	18, 112, 168, 215
Bolles, R. C.	286
Brady, M.	21, 154
Brown, C. M.	79, 115
Bruss, A. R.	204
Bullock, B.	29
Close, D.	29
Davis, L. S.	1, 124
Feldman, J. A.	15
Fischler, M. A.	26, 274, 286, 342
Fouse, S.	42
Hannah, M. J.	283
Hanson, A. J.	342
Hartley, R. L.	323
Herman, M.	179
Hong, T. H.	58, 72
Horn, B. K. P.	154, 204
Kanade, T.	5, 90, 179, 265
Kender, J. R.	265
Kimball, O. A.	79
Kitchen, L.	230, 323
Kuroe, S.	179

AUTHOR INDEX (Cont'd)

SECTION I

PROGRAM REVIEWS
BY
PRINCIPAL INVESTIGATORS

AD#:	TITLE:
AD-P000 102	Understanding Features, Objects, and Backgrounds.
AD-P000 103	Image Understanding Research at CMU.
AD-P000 104	Image Understanding Research at USC:1981-82.
AD-P000 105	Recent Results of the Rochester Image Understanding Project.
AD-P000 106	Geometric Reasoning and Spatial Understanding.
AD-P000 107	MIT Progress in Understanding Images.
AD-P000 108	The SRI Image Understanding Program.
AD-P000 109	IU Application Project: A Progress Report.
AD-P000 110	Radius: A Progress Report.
AD-P000 111	A Survey and Evaluation of FLIR Target Detection/Segmentation Algorithms.
AD-P000 112	Extracting Compact Objects Using Linked Pyramids.
AD-P000 113	Unforced Image Partitioning by Weighted Pyramid Linking.
AD-P000 114	Constraint Interaction in Shape-From-Shading Algorithms.
AD-P000 115	Using Shadows in Finding Surface Orientations.
AD-P000 116	Matching Linear Features of Images and Maps.
AD-P000 117	Representation of Time and Sequences of Events.
AD-P000 118	Modeling The Sequential Behavior of Hough Transform Schemes.
AD-P000 119	Contour-Based Motion Estimation.
AD-P000 120	The Recovery of Surface Orientation from Image Irradiance.
AD-P000 121	Concept Maps.
AD-P000 122	Rotationally Symmetric Operators for Surface Interpolation.
AD-P000 123	Segmentation and Aggregation: An Approach to Figure-Ground Phenomena.
AD-P000 124	Incremental Acquisition of a Three-dimensional Scene Model from Images.

COMPONENT PART NOTICE (CON'T)

AD#:	TITLE:
AD-P000 125	Methods for Interpreting Perspective Images.
AD-P000 126	Passive Navigation.
AD-P000 127	A System for Automated Stereo Mapping.
AD-P000 128	Segmentation In Acronym.
AD-P000 129	Scene Analysis Using Region-Based Constraint Filtering.
AD-P000 130	Structural Texture Analysis Applications.
AD-P000 131	Depth of Scene from Depth of Field.
AD-P000 132	Reasoning between Structure and Function.
AD-P000 133	Gradient Space Under Orthography and Perspective.
AD-P000 134	Linear Delineation.
AD-P000 135	Detection of Errors in Match Disparities.
AD-P000 136	Modeling and Using Physical Constraints in Scene Analysis.
AD-P000 137	Symbolic Matching of Images and Scene Models.
AD-P000 138	Automatic Generation of Depth Maps from Stero Images.
AD-P000 139	Analysis of Low-level Computer Vision Algorithms for Implementation on a VISI Processor Array.
AD-P000 140	Segmentation of FLIR Images: A Comparative Study.
AD-P000 141	The DARPA/DMA Image Understanding Testbed.

17
23
[illegible]

AD-P000 102

UNDERSTANDING FEATURES, OBJECTS, AND BACKGROUNDS
Project Status Report 1 August 1981 - 31 July 1982

Azriel Rosenfeld
Larry S. Davis

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

Current activities on the project are summarized under the following headings:

(a) Preprocessing and segmentation,
(b) Feature detection and texture analysis,
(c) Hierarchical representations,
(d) Matching and motion,

1. Introduction

This project is concerned with the study of advanced techniques for the analysis of reconnaissance imagery. It is being conducted under Contract DAAG-53-76-C-0138 (DARPA Order 3206), monitored by the U.S. Army Night Vision and Electro-Optics Laboratory (Dr. George Jones). The Westinghouse Systems Development Division, under a subcontract, is collaborating on implementation and application aspects.

Work on the current phase of the project was initiated in April 1980. Accomplishments and publications during the period 1 April 1980 - 31 July 1981 are summarized in two earlier status reports [1-2], the first of which also appeared in the Proceedings of the April 1981 Image Understanding Workshop [3]. The present report, covering the period 1 August 1981 - 31 July 1982, is being issued separately and will also appear in the Proceedings of the September 1982 Image Understanding Workshop. For convenience, publications since February 1981 are also cited here, since they were not cited in the April 1981 Workshop Proceedings.

The project is concerned with three principal areas: segmentation techniques; context-based target detection in FLIR imagery; and analysis of time-varying imagery. Work in the first area is summarized in Section 2 (Preprocessing and segmentation) and 3 (Feature detection and texture analysis), while Section 4 summarizes work on the use of hierarchical image representations ("pyramids") in both segmentation and feature detection. Three papers in these areas, dealing with a comparative study of segmentation techniques as applied to FLIR imagery [4], and with the use of pyramids for extracting compact objects from an image [5,6], also appear in the Workshop Proceedings. Work on context-based target detection is covered in a report that also appears in the Workshop Proceedings [7]; a second report on this topic is in preparation. Finally, Section 5 summarizes work on image matching and time-varying imagery analysis; one paper in this area also appears in the Workshop Proceedings [8].

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Clara Robertson in preparing this report.

2. Preprocessing and segmentation

2.1 Comparative segmentation study

A comparative study of FLIR image segmentation techniques was conducted, using a database of 51 images obtained from four different sources. The techniques compared included two- and three-class relaxation, "pyramid linking", and "superspike" (see below). The results are described in detail in [4] and in a paper appearing in the Workshop Proceedings.

2.2 Constraint-based region identification

A context-based approach to region identification on FLIR imagery was developed; it uses constraint filtering techniques to identify regions as (possibly) belonging to the classes sky, smoke, ground, tank, and tree. A detailed description of the approach and examples of its use can be found in [7], which also appears in the Workshop Proceedings.

2.3 Histogram-based image smoothing

A powerful method of edge-preserving image smoothing known as "superspike" has been developed. It is based on repeatedly averaging each pixel with a subset of its neighbors, where the neighbors used are chosen on the basis of their relationships with the given pixel on the image's histogram. Specifically, we use a neighbor if its value is more probable than the pixel's, and there is no concavity on the histogram between its value and the pixel's; these conditions imply that it belongs to the same histogram peak as the pixel, and is higher up on that peak. This method can also be applied to multi-spectral imagery, using the scattergram rather than the histogram [9]. Figure 1 shows an example of this type of smoothing applied to a color image of a house, using only two bands (red and blue). The result is quite cartoon-like, and the scattergram of the smoothed image is virtually reduced to a small set of spikes.

2.4 Segmentation by bimean clustering

The mean is the best-fitting constant, in the least squares sense, to a given set of data. We define the "bimean" of the data as the best-fitting pair of constants. If the data are image gray levels, the bimean defines a segmentation of the levels into two populations, each consisting of those levels that are closer to one of the constants than to the other. An algorithm for finding the bimean of a set of scalar data has been developed. It yields good segmentations in some cases which are not well segmented by the two-class ISODATA clustering algorithm. The details, and examples, can be found in [10].

3. Feature detection and texture analysis

3.1 Edge and corner detection

Hueckel-type edge detectors are based on finding a best-fitting step edge to a given image neighborhood. Some general properties of such detectors have been derived, and applied to defining Hueckel-type detectors for various simple types of neighborhoods. The details are presented in [11].

If an image contains an object on a contrasting background, corners on the object's contour give rise to slope changes in the x- and y-axis projections of the image. Thus detecting such changes indicates which rows and columns of the image are likely to contain corners. The details of the approach, as well as examples, were presented in [12] (also summarized in [2]).

3.2 Texture analysis

A comparative study of texture classification using various types of features was conducted. The best features were (simplified versions of) the "texture energy measures" developed by Laws at USC. The Laws features and texture samples used are shown in Figures 2 and 3, and the results are summarized in Table 1. The details can be found in [13].

Texture analysis methods can be applied to terrain classification using arrays of elevation data, rather than intensity data. Some simple examples and a brief discussion can be found in [14]. This approach will become of increasing interest as high-resolution digital terrain elevation data becomes available over the coming years.

4. Hierarchical methods

A class of methods for image segmentation and object detection has been developed that makes use of a "pyramid" of successively reduced-resolution versions of the image. One such method constructs subtrees of the pyramid representing homogeneous subpopulations of pixels, by creating links between nearby pairs of pixels on consecutive levels of the pyramid based on their similarity in value. This method has been generalized to multispectral imagery, where better results can be obtained using two bands than using one band at a time. The details were given in [15] (also briefly summarized in [2]).

Pyramid linking methods can also be used to extract significant edges from an image, by creating links between nearby pairs of edge segments on consecutive levels based on similarity in slope. The details of this approach were given in [16] (also briefly summarized in [2]).

A more recent application of pyramid linking is to the detection and extraction of compact objects from an image using local "spoke filters" on each level of the pyramid. This method is described in detail in [5], which also appears in the Workshop Proceedings.

Pyramid linking is usually based on forced choices, where a pixel must link to one of the nearby pixels on the level above it. A "softer" approach is to use weighted links (the more similar, the stronger). This too gives rise to trees whose roots are pixels that have only negligibly weighted links to the level above them. Typically, the leaves of such a tree constitute a compact, homogeneous piece of the image. The approach is described in detail in [6], which also appears in the Workshop Proceedings.

5. Matching and motion

5.1 Corner-based image matching

Some experiments on relaxation image matching, based on "corner" features extracted from the images, were described in [17] (also briefly summarized in [2]). Further experiments, in which local gray level correlation was used to resolve ambiguous cases, are described in [18].

5.2 Corner-based motion computation

By computing (approximately) the spatial and temporal derivatives of the image gray level at a given pixel, the component of the velocity of that pixel in the gradient direction can be estimated. If the pixel is at a "corner" of an object, where edges having two different directions meet, its velocity is thus completely determined. When the velocities are due to observer motion ("optical flow"), knowing them at a few points suffices to determine the translation and rotational components of the flow [19]. When an object is moving, estimates of the velocities of its corners can be "propagated" along its contours to yield a consistent estimate of object motion [20,21]. Further details of this approach, together with examples, are presented in [8], which also appears in the Workshop Proceedings.

REFERENCES

1. Understanding Features, Objects and Backgrounds, Semi-Annual Report, 1 April 1980 - 31 January 1981, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, Md.

2. Understanding Features, Objects and Backgrounds, Semi-Annual Report, 1 February - 31 July 1981, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD.

3. Understanding Features, Objects, and Backgrounds, Project Status Report 1 April 1980 - 31 January 1981, Proceedings, DARPA Image Understanding Workshop, April 1981, 208-215.

4. R. L. Hartley, L. J. Kitchen, C. Y. Wang, and A. Rosenfeld, A comparative study of segmentation algorithms for FLIR images, TR-1104, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, September 1981 (an abridged version appears in the Workshop Proceedings).

5. T. H. Hong and M. Shneier, Extracting compact objects using linked pyramids, TR-1123, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, November 1981 (also in the Workshop Proceedings).

6. T. H. Hong and A. Rosenfeld, Unforced image partitioning by weighted pyramid linking, TR-1137, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, January 1982 (also in the Workshop Proceedings).

7. L. Kitchen, Scene analysis using region-based constraint filtering, TR-1150, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, February 1982 (also in the Workshop Proceedings).

8. L. S. Davis, Z. Q. Wu, and H. F. Sun, Contour-based motion estimation, TR-1179, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, June 1982 (also in the Workshop Proceedings).

9. C. Y. Wang and L. Kitchen, Improvements in multispectral image smoothing, TR-1152, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1982.

10. S. Dunn, L. Janos, and A. Rosenfeld, Bimean clustering, TR-1106, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, September 1981.

11. R. B. Boppana and A. Rosenfeld, Some properties of Hueckel-type edge operators, TR-1178, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, June 1982.

12. Z. Q. Wu and A. Rosenfeld, Filtered projections as an aid in corner detection, TR-1078, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, July 1981.

13. M. Pietikäinen, A. Rosenfeld, and L. S. Davis, Texture classification using averages of local pattern matches, TR-1098, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, September 1981.

14. C. Y. Wang and A. Rosenfeld, Elevation texture, TR-1086, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, August 1981.

15. T. H. Hong and A. Rosenfeld, Multiband pyramid linking, TR-1025, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1981.

16. T. H. Hong, M. Shneier, and A. Rosenfeld, Border extraction using linked edge pyramids, TR-1080, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, July 1981.

17. C. Y. Wang, H. F. Sun, S. Yada, and A. Rosenfeld, Some experiments in relaxation image matching using corner features, TR-1071, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, July 1981.

18. H. F. Sun, Image registration by combining feature matching and gray level correlation, TR-1091, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, August 1981.

19. K. Prazdny, Computing motions of (locally) planar surfaces from spatio-temporal changes in image brightness: a note, TR-1090, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, August [illegible]81.

20. L. S. Davis, H. F. Sun, and Z. Q. Wu, Motion detection at corners, TR-1130, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1981.

21. Z. Q. Wu, H. F. Sun, and L. S. Davis, Determining velocities by propagation, TR-1132, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1981.

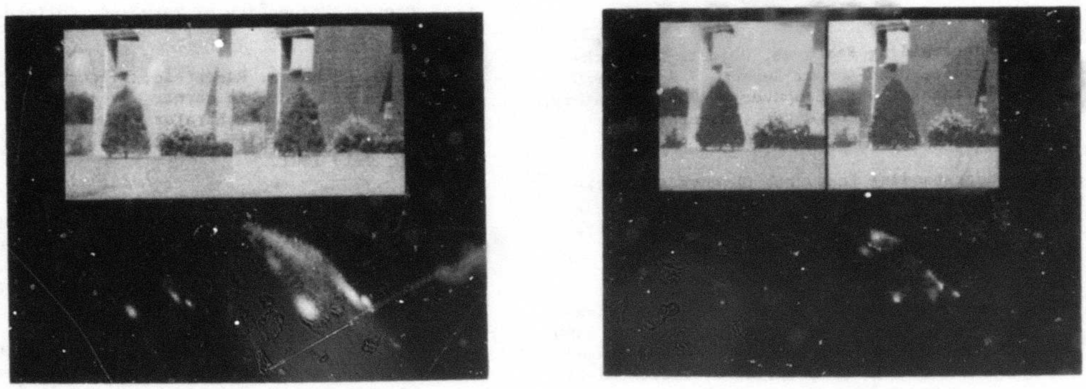


Figure 1. Multispectral "superspike". a) (Top) Red and green bands of a color image of a house. (Bottom) Scatter plot of (red, green) values, linearly (left) and logarithmically (right) scaled.

b) Results after application of "superspike"; the parts correspond to those in (a).

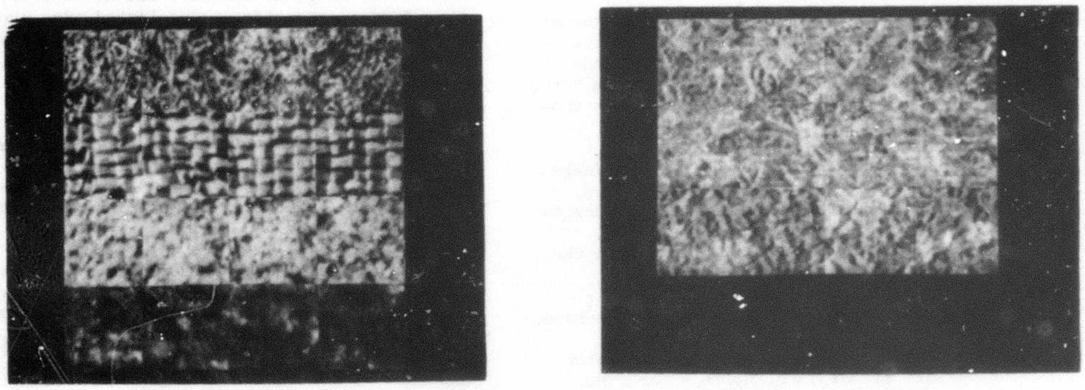


Figure 2. 28 texture samples. Left: grass, raffia, sand, wool. Right: three geological terrain types.

L5E5:

$$\begin{array}{rrrrr} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{array}$$

E5S5:

$$\begin{array}{rrrrr} -1 & 0 & 2 & 0 & -1 \\ -2 & 0 & 4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 0 & -2 & 0 & 1 \end{array}$$

L5S5:

$$\begin{array}{rrrrr} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{array}$$

R5R5:

$$\begin{array}{rrrrr} 1 & -4 & 6 & -4 & 1 \\ -4 & 16 & -24 & 16 & -4 \\ 6 & -24 & 36 & -24 & 6 \\ -4 & 16 & -24 & 16 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{array}$$

Figure 3. Four 5x5 Laws masks.

Feature:	L5E5	E5S5	L5S5	R5R5	CONX	CONY	E/A	WE/A
Score:	23	25	22	25	20	19	19	19

Table 1. Numbers of samples correctly classified using a single texture feature. CONX and CONY are Haralick's CON feature for displacements (1,0) and (0,1); (W)E/A is (magnitude-weighted) amount of edge per unit area.

AD-P000 103

Image Understanding Research at CMU

Takeo Kanade

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

The goals of Image Understanding Research at CMU ~~have been~~ were to develop basic theory for understanding 3-dimensional shapes and to demonstrate an integrated system for photo interpretation (database and interactive/automatic image interpretation techniques). In this report we will present our recent representative progress in the following three subprojects: (1) MAPS; (2) Incremental 3D Mosaic system; and 3) Theory for shape understanding.

MAPS

MAPS (Map Assisted Photo-interpretation System) is intended to be an integrated image/map database system for photo interpretation tasks. We have continued to upgrade MAPS since we reported in the April 1981 proceedings [McKeown, Kanade 81]. As it stands, MAPS is a large, well-developed system.

One of the new modules added is 3D Map display. Using the digital terrain database, it can generate and display a 3D view of an area of interest from a specified view point. Currently the output images are overlayed by color coded thematics which are generated by scan conversion of a polygon map database provided by the Defense Mapping Agency. In addition, detailed cultural features, such as buildings, roads and bridges, can be portrayed in 3D views using the DMA map as a base map. BROWSE [McKeown, Denlinger 82], a window-oriented displayed manager used in MAPS, provides the interface to this capability.

Potential extensions of this module include such capabilities as displaying terrain profiles across a specified path, displaying a natural view by patching (rubber-sheeting) an aerial photo image over the terrain model, and simulating the stream-flow pattern from a specified point.

McKeown [McKeown 82] has also been working on integrating a *concept map* into the system. The concept map database consists of a collection of *concepts* each describing spatial features such as political areas (states, counties), business and residential areas, parks and natural features (rivers, lakes), and man-made features (airports, power stations, universities). The entities of these concepts are hierarchically organized according to both natural geometrical relationships (such as *containment* and *intersection*) and *level-of-detail* relationships. In this way, individual map features can be associated with high-level semantic map descriptions.

Once the concept map is available, since the image-to-map correspondence has been established in the system, MAPS can now provide access to imagery and handle queries to the database through four levels of access:

Signal level	Specify a point or area in the displayed image. The resulting image coordinates are mapped to map coordinates by the image-to-map correspondence, which in turn are used to search the concept map database.
Symbolic level	Specify a symbolic name. A user defined name (e.g., *Memorial Bridge*) is mapped into the map coordinates by the concept map, and these coordinates can be used to access related images.
Role level	Specify roles (e.g., *political*) of interesting concepts. Concepts which satisfy the specification are searched in the concept map database.
Geometric level	Specify the latitude/longitude/elevation. Certain geometric properties such as containment, intersection, and closest-point are computed at this level.

MAPS can now handle queries across all the levels of access in specifying and answering. A few typical examples are:

- "Display images of National Airport before 1976" *(Get image from symbolic level)*
- "What is the closest political building to this [pointing to the

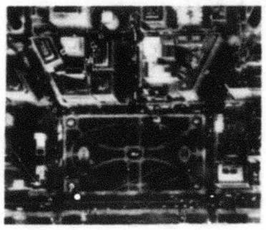


Figure 1: A sample target area

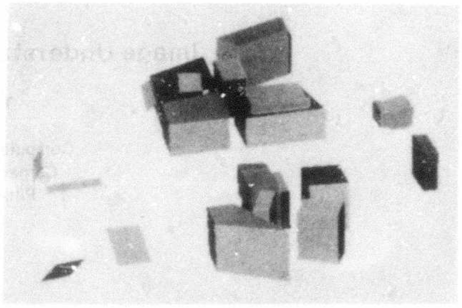


Figure 2: 3D Mosaic flowchart: boxes are major modules and ellipses are data structures

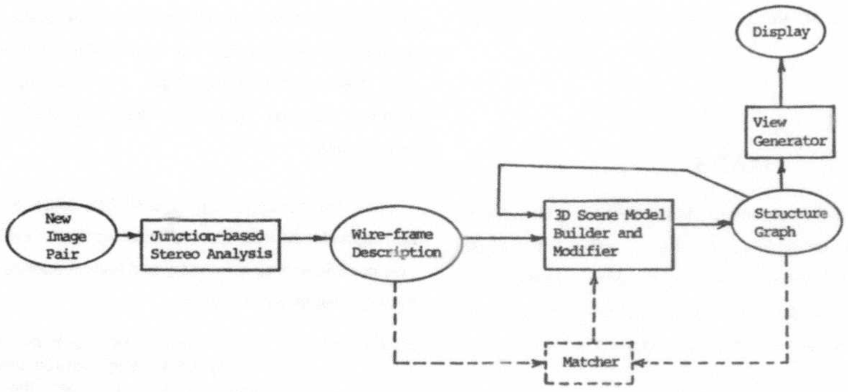


Figure 3: A perspective view of the buildings generated from the 3D scene model that has been constructed by the 3D Mosaic system

three-space direction of illumination is known provides one degree of constraint; light sources whose 3D direction is unknown provide none. However, the shadow geometry shows that shadows provide important benefits for image understanding: shadows allow one to substitute information about the light source position instead of *a priori* knowledge of object surface orientation; shadows allow one to use highly visible shadow edge pairs in place of frequently unreliable edges within shaded portions of an image; increasing amount of information is provided when the shadow falls on many visible, differently oriented surfaces. These are often experienced in photo interpretation.

Some methods have been found for combining shadow geometry with other shape inference methods, such as Horn's shape from shading [Horn 77] and Kanade and Kender's skewed symmetry [Kanade, Kender 80]. Work in progress includes extending the results to perspective images and further exploration of curved surfaces.

screen] geographic point ?" *(Get symbolic level from role and signal levels with geometric constraint)*

- "How many bridges cross the Potomac River between Virginia and the District of Columbia?" *(Get symbolic level from symbolic level with geometric constraint)*

The image/map database is important for photo interpretation because interpretation of a given image is meaningful only in the context of the target area. That context should be quickly provided to the interpreter, and the interpretation result should then be stored in a manner that can be used in the future. Thus an integrated image/map database system, like MAPS, is a vital component in computer photo interpretation systems.

Incremental 3D MOSAIC System

As a step toward automatic interpretation of aerial photos of urban scenes, we have been developing the 3D Mosaic system [Herman, Kanade, Kuroe 82]. The goal of this system is to incrementally acquire a 3D model of a complex urban scene from images. The notion of incremental acquisition arises from the observations that (1) single images contain only partial information about a scene, (2) complex images are difficult to fully interpret, and (3) different features of a given scene tend to be easier to extract in different images because of differences in viewpoint and lighting conditions. Our method involves using multiple views of the scene in a sequential manner. As each successive view is analyzed, the model of the scene is incrementally updated with the information derived from the new view. The model is initially an approximation of the scene, and becomes more and more refined as new images of the scene are acquired and processed.

Currently we are working on a scene in Washington, DC (Figure 1 shows one of the stereo-pair images). The objective is to obtain a 3D scene description of buildings in the area. As shown in Figure 2, the system consists of: a module for stereo analysis, a module for building and modifying the 3D scene model, and a module for generating images from new viewpoints for display purposes. First, stereo analysis based on matching junctions and lines generates a 3D wire-frame description. A surface-based scene model (represented as a structure graph) is then derived from this description. Currently, the model includes only planar surfaces and buildings are approximated by polyhedra. Figure 3 shows a display of the 3D model obtained.

Once the 3D model is built, it can be used for various purposes including: (1) matching the model against new images of the scene for such tasks as model revision and change detection; (2) synthesizing images as seen from hypothetical view points for familiarizing personnel with the area; and (3) using the model for planning paths for flight plans or robot navigation tasks. In this way, the 3D Mosaic scene model plays the role of a central description which (1) reflects the current understanding of the scene, (2) assimilates new information about the scene, and (3) permits decisions dealing with the scene environment to be made.

Theory for Shape Understanding

At CMU, we have been working on fundamental theories, such as the theory of shape from texture [Kender 80] and the theory of mapping image properties into shape constraints [Kanade 81, Kanade, Kender 80], which provide mechanisms to understand 3-dimensional scenes from images. Our emphasis continues to be in the geometrical aspects of image constraints for extracting shape, and we have added new results in this area.

Shadow Geometry

We have been studying the geometric interpretation of shadows in images. Given a line drawing with shadow regions identified and correspondences established with shadow-making regions, Shafer and Kanade [Shafer, Kanade 82] have developed a theory which describes the resulting geometrical constraints upon the orientations of the surfaces involved.

A *Basic Shadow Problem* is first posed, in which there is a single light source, and a single surface casts a shadow on another (background) surface. There are six parameters to determine: the orientation (2 parameters) for each surface, and the direction of the vector (2 parameters) pointing at the light source. It was found that if some set of 3 of these are given in advance, the remaining 3 can then be determined geometrically from the image. (Note that this includes the commonly occurring cases of inferring shape with known sun angle and ground plane.) The solution method consists of identifying "illumination surfaces" consisting of illumination vectors, assigning Huffman-Clowes line labels to their edges, and applying the corresponding constraints in gradient space. A closed-form solution has been found for this problem.

When multiple light sources and multiple surfaces are involved, there is no essential difference. Each light source for which the

Shape Representation

We are exploring the definition, notation, and properties of two types of shape representation: gradient space and generalized cylinders. Shafer, Kanade, and Kender (now at Columbia) [Shafer, Kanade, Kender 82] have compiled a comprehensive presentation of the gradient space, beginning with basic definitions. *Vector gradients* are introduced as a useful addition to surface gradients: the gradient of a vector is the same as the gradient of the surfaces normal to that vector. Vector gradients allow elegant statements about perpendicularity and polyhedral edges in the gradient space.

The relationship between perspective and the gradient space has been fully explored, beginning with definitions of vanishing points and lines, which are very closely related to vector and surface gradients. The *vanishing gradient* of an image line has been defined as the gradient of the surfaces for which that line is the vanishing line. Using vanishing gradients, the connect-edge relationship for perspective has been defined with more precision than in past work.

We have also been studying the fundamental properties of generalized cylinders. Beginning with a formal mathematical definition of generalized cylinders, we have identified several subclasses of particular interest, including those with linear (line segment) axes, those with linear scaling functions, those with cross-sections perpendicular to the axis, and those with circular cross-sections. Formulae for the coordinates of points on the surface of a generalized cylinder and for the surface normals have been derived.

Several interesting theorems have been proven regarding the existence of multiple representations for the same solid shape, and the directions of surface normals at various points on the shape. Current work is centered around the twin problems of predicting the projected shape of a generalized cylinder from various viewpoints, and analyzing an image (silhouette or range data) to deduce as much as possible about the solid shape.

Occluding Boundaries in Computing Optical Flow

We [Cornelius, Kanade 82] have developed an algorithm that assigns velocities to image points by explicitly taking object boundaries into account. The iterative method by Horn and Schunck [Horn, Schunck 81] for computing optical flow (the distribution of apparent velocities of movement of brightness patterns) from an image sequence assumes that the velocities vary smoothly over the entire image. This assumption, however, has limited utility in real images where object boundaries are usually places of velocity discontinuity. The method of Cornelius and Kanade assumes that the velocity and resultant image changes vary smoothly over bounded regions corresponding to objects. It also allows changes in the brightness patterns (due to change in pattern of the object or lighting conditions) so that velocities more closely represent the motions of objects projected onto the image plane.

Discontinuities in velocity which occur at object boundaries must be explicitly accounted for in order to accurately determine velocities within the boundaries. To allow for these discontinuities, the smoothness constraint is applied separately to regions on either side of a boundary. This can be done once the projections of the object boundaries have been located in the image. Interestingly, implementation does not require that the image be segmented into regions corresponding to objects, rather only that the location of *possible* object boundaries be determined.

A change in the brightness pattern refers to the change in image brightness of the same physical point on an object from one frame to the next. This might occur when an object rotates and the lighting hits the object in a different way. For x-ray images of a beating heart, the brightness at a point in the image is dependent on the depth of the heart cavity perpendicular to the image plane. Thus the pattern changes will reflect the expansion or contraction movement of the heart in the direction perpendicular to the image plane. A pattern change will also occur when a point on the object is obscured or revealed in successive image frames. This second type of change causes discontinuities in the velocity across boundaries, which have already been discussed. To allow for pattern changes in the image, the rate of brightness change of a given physical point is treated as another velocity component under the constraint that it changes smoothly within boundaries. While this rate of brightness change is not strictly a velocity, it makes sense to constrain it to vary smoothly within object boundaries, just as is done for the velocity components.

The methods developed are applied to models of ellipsoids and boxes undergoing expansion and rotation, and to x-ray image sequences of a beating heart.

Range Data Acquisition and Analysis

We have developed a laser-scanning ranging device [Kanade, Asada 81], [Kanade, Fuhrman 82]. This device provides range measurement of approximately 0.3 mm resolution in the cubic space of 20x20x20 cm^3 located 60 cm from the sensor; the speed of measurement is 1000 points/sec. It will also simultaneously measure reflectance of surface points.

We are working on techniques for obtaining object-centered descriptions of objects from range data. This bottom-up procedure is essential for tasks such as learning shape by presentation. In working on this problem, Smith [Smith 82] has characterized and classified contours which appear in range images obtained by triangulation. His classification provides insight into the problem common to shadows, stereo, and triangulation-based range finders.

When deducing object geometry from the contours, it is important to understand how reliable they are. However, this topic has not previously been addressed. Indeed, some types of contours which are possible with a triangulation range finder have escaped notice.

The most reliable contours are sharp occluding contours. Receding occluding contours have more uncertainty in position along the line of sight of either the illuminator or camera, whichever has the oblique view. An occluding contour provides information about the object boundary. An occluded boundary, on the other hand, is a projection of an occluding contour onto the occluded object surface along the line of sight of either illuminator or camera: the relationship between occluding and occluded contours is equivalent to that of shadow-making and casted-shadow edges. A problem is to determine which contours are occluding and which are occluded.

Figure 4 depicts a triangulation range finder imaging a sphere and an object behind it. A point on the surface of the sphere can be ranged only when it is visible from both the illuminator and the camera. The deep shadow, which is visible from neither vantage point, is called the *umbra*. The space which is shadowed from one vantage point, but not the other, is called the *penumbra*. A contour separating the rangable face of the sphere from a portion in the penumbra is an *occluding penumbral contour*; a contour separating a section lying in the penumbra from a section lying in the umbra is an *occluding umbral contour*. These classes are further split into subclasses which denote whether the contour is caused by the limit of view of the illuminator or the camera.

Each contour in Figure 4 is labeled with a code. *C* and *I* denote whether it was caused by the vantage point of the camera or the illuminator. *p* and *u* indicate penumbral and umbral contours. *1* and *2* indicate occluding and occluded contours. Thus, a *Cp1* contour arises where an object curves away from the camera, and this contour can cause a *Cp2* contour on an object behind.

The occluding/occluded contour pairs used by Sugihara [Sugihara 79] correspond to *Ip1-Ip2* and *Cp1-Cp2* pairs. An *Ip1-Ip2* pair is easily spotted in the deflection image in the natural (i.e., camera) registration, as it produces a range (deflection) jump. A *Cp1-Cp2* pair can be spotted without conversion to 3D coordinates by reconstructing the image from the illuminator's point of view.

The task of determining which contours are occluded is complicated by the fact that some occluded contours are produced by umbral contours -- yet the umbral contours (along with part of the occluding surface) are missing in the image. There is a gap between the observed occluded edge and the one which would be expected if only the visible surface of the occluding object were casting the shadow. These gaps are the *Iu2-Cx2* and *Cu2-Ix2* portions of the occluded object. This suggests that once we can identify penumbral occluding and umbral occluded contours for both camera and illuminator (i.e., *Cp1, Ip1, Cu2* and *Iu2*) we can infer the shape of a cross section more accurately than simply by fitting a curve to visible contours [Agin, Binford 73].

Although one cannot be certain whether a contour is an occluding contour or a *u2* contour, there are some necessary conditions:

- An *Iu2* contour occurs only when the deflection value undergoes a drop between the left and right sides of a validity gap in the natural registration.
- A *Cu2* contour occurs only when the deflection value undergoes a drop between the left and right sides of a validity gap in the camera registration.

Smith has also developed some additional heuristics which help in identifying contour types. His algorithms for detecting and classifying contours have been successfully tested with real range data of complex scenes. Work is in progress to obtain shape representation of objects in the scene for recognition.

Image and Feature Matching

Lucas has worked on an image registration algorithm based on linear approximations of the image at each point [Lucas, Kanade 81]. In these proceedings, Lucas [Lucas 82] shows how the algorithm can be augmented by smoothing the images, iterating the calculation, and using appropriate weighting factors. A theoretical analysis concerning the performance of the algorithm (efficiency and convergence) has also been performed.

Testbed Contribution

CMU has made the following contribution to the Testbed being constructed at SRI:

CMU image access and Grinnell Graphics packages
: After modification, these became the bases of the Testbed basic packages.

PHOENIX region segmentation program
: This is an Ohlander-Price-Shafer region segmentation program with a flexible user interface and interactive graphics.

Improved Moravec stereo
: This is a C version of the Moravec stereo reconstruction algorithm. Several improvements were added to increase speed and flexibility.

References

[Agin, Binford 73] Agin, G.J. and Binford, T.O.
Computer Description of Curved Objects.
In *Proc. 3rd International Joint Conference on Artificial Intelligence*, pages 629-640. Aug., 1973.

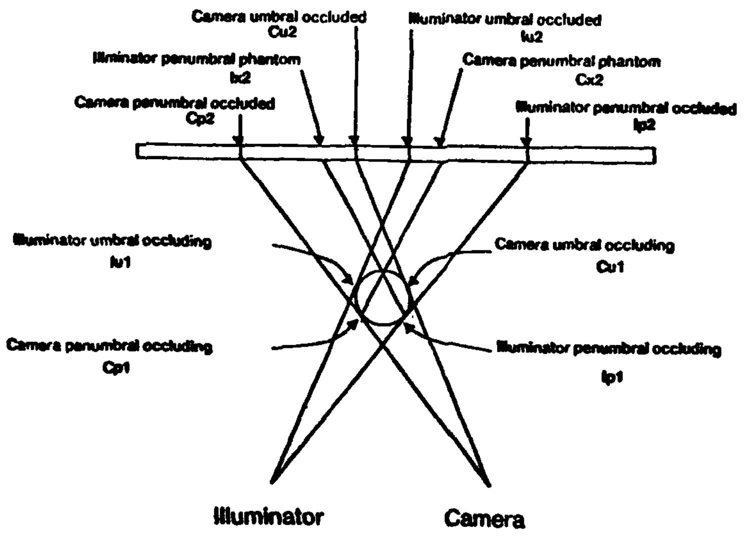
[Cornelius, Kanade 82]
Cornelius, N. H. and Kanade, T.
Computing 2D Velocity Field Using Occluding Boundary Information.
Technical Report (in preparation), Carnegie-Mellon University, Computer Science Department, 1982.

[Herman, Kanade, Kuroe 82]
Herman, M., Kanade, T. and Kuroe, S.
Incremental Acquisition of a Three-dimensional Scene Model from Images.
In *(this volume)*. 1982.

[Horn 77] Horn, B. K. P.
Understanding Image Intensities.
Artificial Intelligence 8:201-231, 1977.

[Horn, Schunck 81]
Horn, B. K. P. and Schunck, B.
Determining Optical Flow.
Artificial Intelligence 17:185-203, 1981.

[Kanade 81] Kanade, T.
Recovery of the Three-Dimensional Shape of an Object from a Single View.
Artificial Intelligence 17:409-460, 1981.



Camera umbral occluded
Cu2
Illuminator umbral occluded
Iu2
Illminator penumbral phantom
Ix2
Camera penumbral phantom
Cx2
Camera penumbral occluded
Cp2
Illuminator penumbral occluded
Ip2
Illuminator umbral occluding
Iu1
Camera umbral occluding
Cu1
Camera penumbral occluding
Cp1
Illuminator penumbral occluding
Ip1
Illuminator
Camera

Figure 4: Types of contours in a triangulation-based active-illumination range finder

[Kanade, Asada 81]
Kanade, T., Asada, H.
Noncontact Visual 3-D Sensing Devices.
In *SPIE Technical Symposium East '81, 3-D Machin Perception (Vol. 283)*. Society of Photo-Optical Instrumentation Engineers, 1981.

[Kanade, Fuhrman 82]
Kanade, T. and Fuhrman, M.
A Laser-Scanning Range Finder for Robotic Application.
Technical Report (in preparation) 1982.

[Kanade, Kender 80]
Kanade, T. and Kender J. R.
Mapping Image Properties into Shape Constraints: Skewed Symmetry, Affine Transformable Patterns and the Shape-from-Tecture Paradigm.
Technical Report CMU-CS-80-133, Carnegie-Mellon University, Computer Science Department, July, 1980.

[Kender 80] Kender, J. R.
Shape from Texture.
PhD thesis, Carnegie-Mellon University, Computer Science Department, 1980.

[Lucas 82] Lucas, B. D.
Automatic Generation of Depth Maps from Stereo Images.
In *(this volume)*. 1982.

[Lucas, Kanade 81]
Lucas, B. D. and Kanade, T.
An Iterative Technique of Image Registration and Its Application to Stereo.
In *Proc. 7th International Joint Conference on Artificial Intelligence*, pages 674-679. Aug., 1981.

[McKeown 82] McKeown, D. M. Jr.
Concept Maps.
In *(this volume)*. 1982.

[McKeown, Denlinger 82]
McKeown, D. M. Jr. and Denlinger, J. L.
Graphical Tools for Interactive Image Interpretation.
In *SIGGRAPH '82 (Computer Graphics Vol. 16, No. 3)*, pages 189-198. July, 1982.

[McKeown, Kanade 81]
McKeown, D. M. Jr. and Kanade, T.
Database Support for Automated Photo Interpretation.
In *Proc. DARPA Image Understanding Workshop*, pages 7-13. April, 1981.

[Shafer, Kanade 82]
Shafer, S. and Kanade, T.
Using Shadows in Finding Surface Orientations.
Technical Report CMU-CS-82-100, Carnegie-Mellon University, Jan., 1982.

[Shafer, Kanade, Kender 82]
Shafer, S., Kanade, T. and Kender, J.
Gradient Space Under Orthography and Perspective.
Technical Report CMU-CS-82-123, Carnegie-Mellon University, May, 1982.

[Smith 82] Smith, D. R.
Analysis of Rangefinder Imagery.
Thesis Proposal, Computer Science Department, Carnegie-Mellon University.

[Sugihara 79] Sugihara, K.
Range-data Analysis Guided by a Junction Dictionary.
Artificial Intelligence 12:41-69, 1979.

[illegible]

IMAGE UNDERSTANDING RESEARCH AT USC:1981-82

R. Nevatia
Departments of Electrical Engineering
and Computer Science
University of Southern California
Los Angeles, California 90089-0272

1. Introduction

This paper summarizes our major research activities since the last Image Understanding Workshop in April 1981. More details can be found in four other detailed technical papers describing our work in these proceedings [1-4]. Our research goals have been to develop a set of general techniques that can be applied to a number of application tasks. Our focus at the high level has been on symbolic matching of an image to a map or another image. This task is central to many applications, e.g., image based navigation, map-updating, and change detection. Our other research has been in support of this high level goal and has included work on segmentation, texture analysis, object detection and description. We have also worked with Hughes Research Laboratories in defining and developing special purpose hardware for IU tasks.

2. Image to Map Correspondence

We have developed two major systems for this task. The first matches line segments found in an image with corresponding lines in the model (which may be a map or another image). This method is applicable when geometric distortions are small and precise positions of lines or edges can be used for matching. It is fast and tolerant to local errors. When a line segment in the image corresponds to a segment in the model (or another image) the location of all other segments are restricted to limited areas with restricted orientations. A small subset of the lines from one image are used in an initial relaxation based scheme which tests pairwise matches using the above constraint. Given this initial kernel all other segments can be quickly matched or discarded as unmatched using the constraints given by the kernel. Details of this method and some results are given in [1].

The second matching system is the result of long term work in general symbolic analysis and has depended on development of other image analysis techniques such as high performance line finders and general segmentation techniques. This method matches a network descriptions derived from extracted lines and regions and their relationships. We have presented the results of this system at previous IU workshops [5]; our recent work uses groups of features in a relaxation matching system and is described in detail in another paper in these proceedings [2].

3. Symbolic Texture Analysis

We have been developing a system for symbolic descriptions of textures; the descriptions are in terms of primitives and their arrangement. The basic approach is to find micro-edges and search for repetitive patterns among them. This process gives us the periodicity of the texture, if any, and the dominant size of the elements, if any. Next, the complete primitives are isolated and then their geometrical arrangements computed. The symbolic descriptions are sufficient to reconstruct regular textures.

We have also applied these descriptions to the recognition of natural textures, commonly used in texture work, such as wool, water, raffia, grass, etc., and achieved a better than 90% accuracy. The errors are between similar, random textures and could be easily reduced if metric information (size) were used. We have also applied these descriptions to estimating 3-D surface orientations using texture gradients.

Our basic description technique has been presented at a previous IU workshop [6]; the new work including surface orientation estimation is presented in another paper here [3]. Complete details may be found in a USC Ph.D. thesis [7].

4. Object Detection

We have made a start on detecting 3-D objects based on analysis of shadows they cast. Figure 1 shows part of an aerial image with some buildings and their shadows. The buildings may be hard to distinguish from other similar shaped structures, e.g., a parking lot, without shadows. We are able to correspond shadows and objects under certain simplifying assumptions: a known distant source of light, vertical objects seen straight down (vertical sides not seen), flat and level ground, and the object shape being generically known (a combination of rectangles in this case).

Our basic approach is to extract line segments and corners first (see Figs. 2 and 3 for example) and to characterize the corners as being likely to belong to a building or not. This is based on the corners having the appropriate photometric properties (brighter than surround) and also on being able to find a matching shadow corner. The evidence from the individual corners is combined when the corners share a segment. If a closed boundary is found, it is examined for consistency of shape. Else, the partial boundary serves as a guide for searching for potential missing parts. Figure 4 shows the building like structures extracted in Fig. 1. The top building was found as a closed boundary by the program. The lower one required searching for nearby segments and connecting two halves because they compliment each other according to the program's predictions. In addition to the segments, we are also able to determine the (relative) heights of the buildings. The details of this process may be found in our recent progress report [8].

5. Hardware Development

In work with Hughes Research Laboratories, we have continued the development of the RADIUS processing system. RADIUS is capable of performing a variety of programmable operations, such as convolution and statistical moments. The current implementation uses a 5x5 kernel, but is modular and expandable to larger sizes. The RADIUS processor has been constructed and tested. It is to be interfaced to a PDP-11 via a UNIBUS, allowing further processing on a general purpose machine.

The choice of the operation for the RADIUS processor was based on an analysis of three low-level processing systems: Nevatia-Babu line finder [9], Law's "texture energy" measure [10] and Ohlander-Price region segmentation [11].

We have also begun a study of the processing required to perform operations after convolution, e.g., thinning, linking, shrink and expand. We believe that such operations can be performed by a single, programmable, grey-level "logic processor."

Details of the hardware implementation effect may be found in [4], in these proceedings.

6. References

1. G.G. Medioni, "Matching Images and Maps," Proc. of the DARPA Image Understanding Workshop, Palo Alto, Ca., September 1982 (these proceedings).

2. K. Price, "Relaxation Matching of Images and Scene Models," Proc. of DARPA Image Understanding Workshop, Palo Alto, Ca., September 1982 (these proceedings).

3. F. Vilnrotter, R. Nevatia, and K. Price, "Structural Texture Analysis: Summary and Applications," Proc. of DARPA Image Understanding Workshop, Palo Alto, Ca., September 1982 (these proceedings).

4. G. Nudd and S. Fouse, "Radius: A Progress Report," Proc. of DARPA Image Understanding Workshop, Palo Alto, Ca., September 1982 (these proceedings).

5. K.E. Price, "Relaxation Matching Applied to Aerial Images," Proc. of DARPA Image Understanding Workshop, Washington, D.C., April 1981, pp. 22-24.

6. F. Vilnrotter, R. Nevatia, and K.E. Price, "Structural Analysis of Natural Textures," Proc. of DARPA Image Understanding Workshop, Washington, D.C., April 1981, pp. 61-68.

7. F. Vilnrotter, "Structural Analysis of Natural Textures," USC Report ISG 100, September 1981.

8. A. Huertas, "An Edge Based System for Detecting Buildings in Aerial Images," in "Image Understanding Research," R. Nevatia, (Ed.), USC Report ISG 101, March 1982.

9. R. Nevatia and K.R. Babu, "Linear Feature Extraction and Description," Computer Graphics and Image Processing, Vol. 13, pp. 257-269.

10. K.I. Laws, "Textured Image Segmentation," USC Report IPI 940, January 1980.

11. R. Ohlander, K. Price, and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Computer Graphics and Image Processing, 1978, pp. 313-333.

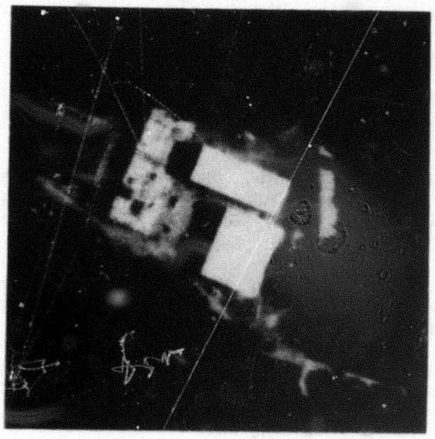


Figure 1. Image

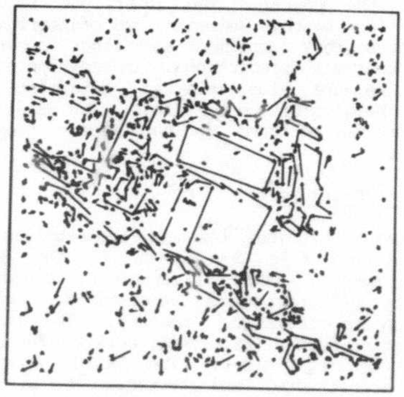


Figure 2. Edge segments

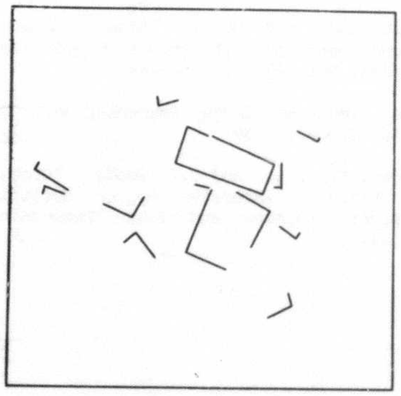


Figure 3. Corners

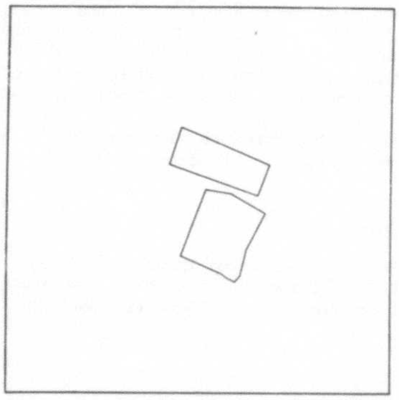


Figure 4. Boxes

AD-P000 205

Recent Results of the Rochester Image Understanding Project

J.A. Feldman

Contents:

University of Rochester
Rochester, New York 14627

1. Robust Vision Operators

1.1. Parameter Networks and the Hough Transform

One of the most difficult problems in vision is segmentation. Recent work has shown how to calculate intrinsic images (e.g., optical flow, surface orientation, occluding contour, and disparity). These images are distinctly easier to segment than the original intensity images. Such techniques can be greatly improved by incorporating Hough methods. The Hough transform idea has been developed into a general control technique. Intrinsic image points are mapped (many to one) into 'parameter networks' [Ballard, 1981]. This theory explains segmentation in terms of highly parallel cooperative computation among intrinsic images and a set of parameter spaces at different levels of abstraction.

The most recent application of these ideas are to improved shape-from-shading calculations [Brown et al., 1982] and motion extraction [Ballard & Kimball, 1982]. Both of these domain specific efforts are closely linked to our new work on a more general theory of Hough-like computations and general implementation techniques for them. Hough-like computations may be modeled as a form of imaging [Brown 1982]. The resulting insights suggested the CHough technique to sharpen peaks and reduce bias in Hough accumulator space [Brown & Curtiss, 1982].

The theory is also useful in analysis of cache-based Hough Transform implementations. It is an appealing idea to use a small content-addressable store to accumulate Hough transform results, rather than a potentially huge multi-dimensional array. Several technical issues are involved in any such scheme, but the idea appears quite promising [Brown & Sher, 1982].

1.2 Adaptive Operators

Control is a crucial issue in Image Understanding. We have been investigating the role of low-level adaptive operators in both the analysis of aerial images and in problem solving. Early aerial image work was reported in the last IU proceedings [Selfridge and Sloan, 1981].

In general, problem solvers cannot hope to create plans that are able to specify fully all the details of operation beforehand and must depend on run-time modification of the plan to insure correct functioning. Fortunately, many primitive actions are highly stereotyped and can be performed by adapting pre-programmed tactics to the current goal context and operating environment. The recently completed thesis of Selfridge [1982] shows how this idea can be applied to fairly difficult problems in aerial image understanding. Related efforts are underway in robot constrution tasks and intelligent interfaces to distributed systems.

1.3 Medical Applications

A system has been built in which Computer Tomograms of the human abdomen are searched as a 3-D image and matched against a detailed geometrical model of the abdomen anatomy. Detected organ boundaries serve to construct an instance of the model that reflects the actual anatomy of a particualr patient as revealed by the corresponding image data. The model-directed approach makes possible the detection of hard-to-find organs (e.g., kidneys) based on known locations of easy-to-find organs (e.g., spinal column), thus relaxing the problem of obscured boundaries in noisy data that tend to hinder data-directed approaches. The model is hierarchical, built of generalized cylinders, and is inherently parallel. It captures relational, structural, and quantitative knowledge that is represented as both data and procedures [Shani, 1981].

Work was also recently completed on a system for reconstructing the shape of the human heart from ultrasound data [Schudy, 1982]. Our work on the automated detection of possible tumor sites in chest radiographs has reached the stage of extensive testing. All of these efforts are contributing both specific and general techniques to our DARPA tasks.

2. Computing with Connections

There is a rapidly growing interest in problem-scale parallelism, both as a model of animal brains ans as a pradigm for VLSI. Work at Rochester has concentrated on connectionist models and their application to vision. The framework is built around computational modules, the simplest of which are termed p-units. We have developed their properties and shown how they can be applied to a variety of problems [Feldman & Ballard, 1982]. More recently, we have established powerful techniques for adoption and change in these networks [Feldman, 1982].

One view of this work is that it extends our development of robust Hough-like operators to more complex IU tasks. A major milestone was achieved with Sabbah's thesis on massively parallel recognition of Origami-world objects [Sabbah, 1982]. Sabbah's work extended the connectionist methodology to a problem domain with several hierarchical structural levels. The resulting program is, to our knowledge, the most noise-resistant system for dealing with this level of complexity. Figure 1 shows the program converging to the correct reading of a self-occluding object. One outcome of Sabbah's effort is the start of a project to build a general purpose simulator for massively parallel systems [Shastri et al., 1982].

3. Shape

The description and recognition of complex shapes continues to be a major focus of the project. The analysis of the dot product space representation has been improved to handle certain pathological cases, and has been generalized to accommodate different criteria for the goodness of the representation.

This simple concept of shape has been applied to the problem of reconstructing three-dimensional surfaces from very sparse data. The key idea is to use appropriate shape descriptors to hypothesize a transformation which accounts for the difference in shape between successive contours. When the hypothesized transformation is minor, very simple-minded surface reconstruction techniques are sufficient. When there are major differences in shape or position between successive contours, our method hallucinates new contours, using the hypothesized shape transformation [Sloan and Hrechanyk, 1981].

More recent efforts treat shape within the connectionist framework described above. Hierarchical descriptions of shapes were considered in [Ballard & Sabbah, 1981]; A major current project involves developing techniques for recognizing articulated objects, whose shape is subject to change [Hrechanyk & Ballard, 1982].

4. General Theory of Vision

Work in our laboratory, among others, has demonstrated strong links between powerful IU techniques and computations used by animal visual systems. We have established strong ties with a wide range of visual scientists at Rochester and a variety of collaborative efforts are underway. One early project is to survey the computational similarities in natural and computer vision [Ballard & Coleman, 1982]. Another effort is our attempt to develop a general framework for theories of vision that would provide a common structure for integrating studies from various disciplines [Feldman, 1982].

References

Ballard, D.H., "Parameter Networks: Towards a Theory of Low-Level Vision," *Proceedings*, 7th IJCAI, Vancouver, British Columbia, August 1981.

Ballard, D.H. and P.D. Coleman, "Cortical Connections: Structure and Function," forthcoming Technical Report, Computer Science Department, University of Rochester, August 1982

Ballard, D.H. and O.A. Kimball, "Rigid Body Motion from Depth and Optical Flow," submitted to *CGIP* Special Issue on Computer Vision: also TR70, Computer Science Department, University of Rochester, November 1981.

Ballard, D.H. and D. Sabbah, "Detecting Object Orientation from Surface Normals," *Proceedings*, 7th IJCAI, Vancouver, British Columbia, August 1981.

Brown, C.M., "Bias and Noise in Hough Transform: Theory," TR105, Computer Science Department, University of Rochester, July 1982.

Brown, C.M. and M. Curtiss, "Bias & Noise in Hough Transform: Experiments," TR113, Computer Science Department, University of Rochester, August 1982.

Brown, C.M. and D. Sher, "Modeling the Sequential Behavior of Hough Transform Schemes," TR114, Computer Science Department, University of Rochester, August 1982; also *Proceedings*, DARPA Image Understanding Workshop, November, 1982.

Feldman, J.A., A Connectionist Model of Visual Memory, in *Parallel Models of Associative Memory*, G.E. Hinton and J.A. Anderson (eds.), Hillsdale, NJ: Lawrence Erlbaum Associates, publishers, 1981.

Feldman, J.A., Memory and Change in Connection Networks, TR96, Computer Science Department, University of Rochester, December 1981.

Feldman, J.A., Four Frames Suffice: A Provisionary Model of Vision and Space, TR99, Computer Science Department, University of Rochester, in preparation.

Feldman, J.A. and D.H. Ballard, Connectionist Models and their Properties, to appear, *Cognitive Science*, 1982.

Hrechanyk, L. and D.H. Ballard, "A Connectionist Model for Shape Perception," *Proceedings*, Workshop on Computer Vision: Representation and Control, Rindge, New Hampshire, August 1982.

Sabbah, D., A Connectionist Approach to Visual Recognition, TR107, Computer Science Department, University of Rochester, April 1982; also Ph.D. thesis, Computer Science Department, University of Rochester, April 1982.

Schudy, R.B., Harmonic Surfaces and Parametric Image Operators: Their Use in Locating the Moving Endocardial Surface from Three-Dimensional Cardiac Ultrasound Data, Ph.D. thesis, University of Rochester, April 1982.

Selfridge, P.G., Reasoning About Success and Failure in Aerial Image Understanding, TR103, Computer Science Department, May 1982; also Ph.D. thesis, Computer Science Department, University of Rochester, December 1981.

Selfridge, P.G. and K.R. Sloan, Jr., Locating Objects Under Different Conditions: An Example in Aerial Image Understanding, *Proceedings: Pattern Recognition and Image Processing*, Dallas, Texas, August 1981.

Selfridge, P.G. and K.R. Sloan, Jr., Reasoning About Images: Application to Aerial Image Understanding, *Proceedings*, Image Understanding Workshop, April 1981.

Shani, U., Understanding Three-Dimensional Images: The Recognition of Abdominal Anatomy from Computer Axial Tomograms (CAT), TR82, Computer Science Department University of Rochester, August 1981.

Shastri, L., M. Brucks, and S. Small, "ISCON: A Network Construction Aid and Simulator for Connectionist Models," TR109 Computer Science Department, University of Rochester, September 1982.

Sloan, K.R., Jr. and L. M. Hrechanyk, Surface Reconstruction from Sparse Data, *Proceedings: Pattern Recognition and Image Processing*, Dallas, Texas, August, 1981.

Sloan, K.R., Jr., Dynamically Quantized Pyramids, *Proceedings*, 7th IJCAI, Vancouver, British Columbia, August 1981.

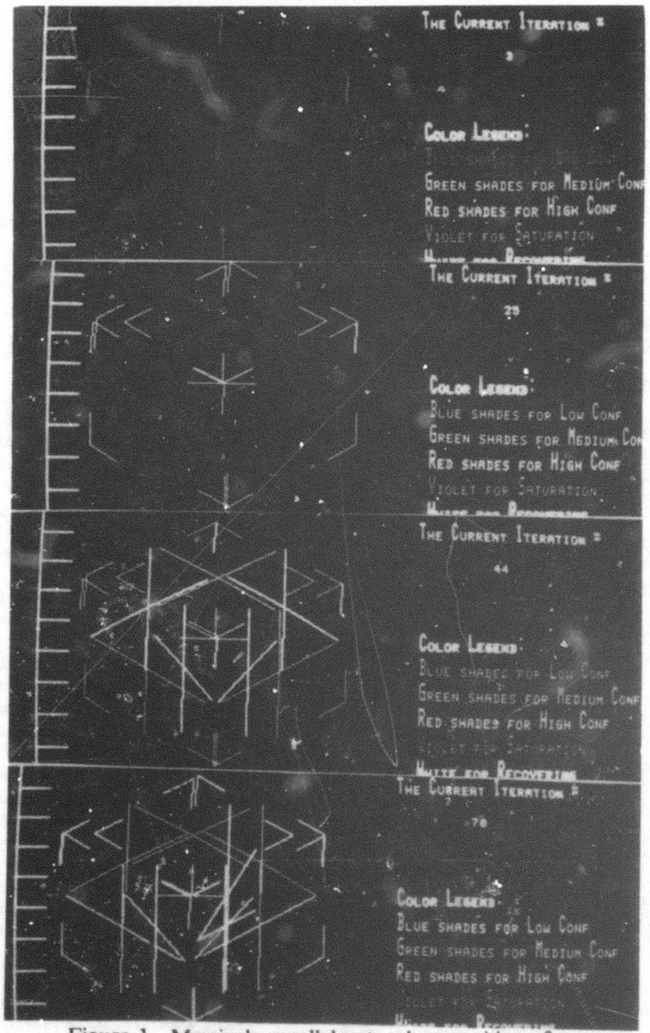


Figure 1 Massively parallel netv.ork recognition of open box from edge data. The extra lines depict levels of abstraction successively developed by the sytem. [cf. Sabbah 1982].

AD P000 106

GEOMETRIC REASONING AND SPATIAL UNDERSTANDING

Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

Progress has been made on extensions to ACRONYM which include: representation and reasoning with time, events, and sequences; collaboration with MIT to develop geometric learning; representation of function, and reasoning between structure and function. A new ribbon finder for ACRONYM is under construction. Work in figure/ground separation is underway as a basis for the ribbon finder. Preliminary results are shown in grouping operations to determine regularities in images. A stereo system has been completed which combines edge-based stereo matching with surface interpolation utilizing correspondence of gray levels. Design of a new stereo vision system is underway.

Introduction

The ACRONYM system is in use by Hughes in an Image Understanding project. ACRONYM has been transported to the VAX by Brooks, and now runs on the Testbed, without the ribbon finder.

We see the need in applications of ACRONYM to interpret time-varying phenomena, to identify coherent sequences of events, to integrate space/time information in interpretation. We have added a new capability to ACRONYM to represent and to reason with time, events, and sequences [Malik 82].

Efforts are underway to simplify the programming of applications in ACRONYM. We have begun a collaboration with MIT to develop geometric learning for vision to incorporate in ACRONYM. Initial results are reported in [Binford 82a]. A central issue in learning is representation. The key theme here is function = structure, which provides a solution to the problem of representing function. Function is represented by structure in space/time. [Lowry 82] discusses reasoning between structure and function. We have also begun defining a task specification language for ACRONYM.

In previous tests of ACRONYM's interpretation of aircraft in images, its performance was limited by the quality of the input data it received, ribbons from a goal-directed ribbon finder [Brooks 80]. [Marimont 82] describes progress on a new ribbon finder for ACRONYM, based on an improved edge finder. The ribbon finder segments edges into curved segments which are cubic splines.

One paradigm for interpretation in ACRONYM has been model-based, goal-directed vision. ACRONYM selects candidates in images and tests them against three-space models. To select candidates, ACRONYM matches image predictions with image observations. ACRONYM has mechanisms to make quite general predictions; it predicts quasi-invariants in images, it predicts appearances of features rather than total views (for a polyhedron of n faces, there are 2^n total views), its predictions are symbolic expressions with variables. Although ACRONYM does it about as well as can be expected, interpretation which is based on prediction of images has limited generality.

The other paradigm for interpretation in ACRONYM is constructive inference. This approach is the focus of our work. This form of interpretation is essential for a general vision system, although it is much more difficult than the model-directed approach. Powerful mechanisms for general interpretation of images as surfaces in three-space were described in [Binford 81] and [Lowe 81]. These procedures interpret local three-dimensional cues given by image curves. [Lowe 82] and [Binford 82b] extend this work in several ways, centered on the theme of figure/ground separation. One extension is to consider inference of grouping processes in the image, without a clear three-dimensional interpretation. The problems of finding clusters of dots, edges, and texture groupings are included. The other extension is to include a new class of cues to grouping image curves into objects, in forming ribbons and grouping ribbons.

Continued research in stereo vision has been sponsored in part by this IU program, augmenting support by RADC. [Baker 82] reports on a system which combines edge-based stereo matching with surface interpolation utilizing correspondence of gray levels. Members of the IU group have made an extensive critical survey of stereo vision systems and the supporting vision technology [Binford 82c]. Design of a new stereo vision system is underway.

A central problem in machine vision is implementing vision algorithms in real time, that is, fast enough for requirements of applications. [Miller 82b] describe architecture studies aimed at development of an array of 128x128 processors in wafer scale integration. They demonstrate an algorithm for routing the array around defective elements, an algorithm effective for defective processor rates less than 10%. They describe software experiments used to explore issues of architecture.

Geometric Reasoning

The representation of time assumes events and an observer with an internal clock, i.e. a mechanism for judging before and after, with an internal metric. Semantically related clusters of events have local time frames [Malik 82]. Coordinate transformations relate time frames. Events may be continuous or discrete. The representation of time makes use of the same mechanisms as the representation of space. Parallelism and synchronization are represented by constraints. Thus, the ACRONYM constraint manipulation system provides most of the mechanism for representing and manipulating constraints. Constraints are linear symbolic expressions with variables. Constraint resolution becomes a linear programming problem which is solved by a simplex algorithm. The method of testing satisfiability of constraints is complete. Upper and lower bounds can be determined. The uniformity of time and space representation makes it attractive to consider space and time together, incorporating special relativity effects, for example.

Objects in ACRONYM are defined by generic functional classes, not by their geometric form [Binford 82d]. However, only generic geometric classes have been implemented. We define object classes by function, but we identify objects visually by their geometric structure. [Lowry 82] describes reasoning between structure and function. Structure is represented by kinematic and dynamic relations among geometric forms represented as generalized cylinders. A key issue is representation of function. The relation form = function provides a strong argument for representing function of most objects in terms of abstract structures, i.e. by abstracted forms of kinematic and dynamic relations among abstracted geometric forms. The mechanism of abstraction is based on partially-specified generalized cylinders [Binford 82d]. These mechanisms require the introduction of physics knowledge which presupposes the introduction of time.

Introduction of learning in ACRONYM has a strong practical focus for applications. Programming of applications requires the building of geometric models and the input of background knowledge including knowledge of experts in image analysis. The knowledge base required for general vision systems can be very large. Our success in simplifying input to vision systems rests on success in non-trivial structural learning, i.e. learning about objects, relations, and programs. An essential part of learning is relating function and form; since fundamental definitions of object classes are functional, this means defining the form of object classes as abstract shapes. This research is intended to provide effective mechanisms for "letting the machine do the work" in building large data bases required for interpretation. There has been much talk and little research in this area; unrealistic hopes should not be raised about when these capabilities will be available. Learning of data is more usual; learning of procedures is central to expert systems for image understanding. There are two themes to this approach to learning. The first theme is specialization to structure and function. We believe that a functional understanding of a system is an economical way to understand a great body of structural information about the system. The utility of this idea depends on whether it is simpler to build a data base of functional information, or to input directly the structural information needed for interpretation. If functional information is compact in useful domains, this approach will succeed. We are convinced that it will succeed. For example, knowing the dimensions of the human body constrains the dimensions of most cultural articles used by individuals, like desks, chairs tables, passenger compartments of cars, cups, etc. Physical considerations constrain transportation objects, like shape of aircraft. Cost and construction constrain shape, size, and materials.

The second theme is learning causal or criterial relations in distinction to statistical correlations, because causal relations define decision criteria which are dependable. In contrast, decisions based on statistical distributions are often unreliable because they assume random samples and typically have systematic bias. It is usually convenient to present biased samples for teaching, in fact is quite difficult to avoid bias in selecting samples of data, and difficult to obtain large samples of data. For example, in locating airfields, the length of an airfield is determined by the takeoff and landing distance of aircraft which it serves. Knowledge of function of the airfield, which aircraft it serves, allows putting strong rather than weak constraints on the length of runways. Initial work involves using ACRONYM for geometry, combined with a system for reasoning by analogy by Winston, and a natural language input system by Katz. Whether it will be feasible to integrate these systems remains to be seen.

Segmentation

[Baker 82] describes a stereo system which is primarily edge-based, but uses image intensities for interpolation of surfaces between edges. The system uses constraints from [Arnold 80] along epipolar lines in a Viterbi search procedure to find best correspondence on a line-by-line basis. Combinatorics of the solution are reduced by using a coarse-fine search procedure, as introduced by [Moravec 1977]. Continuity of edges between epipolar lines provides a strong constraint on solutions, as demonstrated by [Arnold 77].

In support of research on stereo vision, a survey of stereo vision systems and supporting vision technology was conducted. Of course, vision mechanisms relevant to stereo include nearly all of vision. Thus, the scope of the study was very broad [Binford 82c]. The survey took the form of a set of topics, each with a critical overview summarizing the state of the art and the principal important ideas with general value. In each topic were a set of critical reviews of major papers. Over two hundred papers were reviewed. A large bibliography was assembled.

We are working to build an advanced stereo system integrating constraints at all levels in a rule-based system.

ACRONYM interprets images at the level of ribbons, matching predicted ribbons with those observed. An improved ribbon finder is under construction [Marimont 82]. The ribbon finder has several components, edge finding based on lateral inhibition, curve linking, curve segmentation, and grouping. The edge finding module uses lateral inhibition to remove the effects of smooth shading. The module detects edges as above-threshold gradients in the signal after lateral inhibition. Edges are localized by the maximum of the gradient, equivalent to the zero crossing of the second derivative of the signal after lateral inhibition [Binford 81]. Edges are linked in four neighbor connectedness in a single raster scan through an image. Curve segmentation is performed at extrema of curvature of linked curves. The representation of curves can be regarded as splines for which knots are well-chosen, i.e. knots are chosen at discontinuities. Splines may have discontinuities in position and/or tangent at corners. Other knots are added as necessary; at these knots, splines are continuous in position and tangent. Cubic splines are used as a basis. The forming of ribbons is not yet complete. It is based on a few fundamental operations including continuity of curves and translational invariance from [Nevatia 74].

We are conducting research to build a fundamental underpinning for mechanisms of ribbon finding and related grouping operations. Now ACRONYM matches single ribbons, then pairs of ribbons. This process is combinatorial. Pairing of edges in ribbons allows many potential combinations, e.g. lanes of a highway. We seek to cut these combinatorics of matching by selecting groups to match which are well-chosen. One aspect of the problem is determining what image structures provide evidence of objects in three-space. This is related to the classical figure/ground problem. It is considered as an extension of inference rules for interpreting images as surfaces [Binford 81]. Ribbons are an example of area structures; they appear as projections of generalized cylinders. Junctions or stars composed of curves or of ribbons are another class of image structures. Even though the rules with vertices have wide application and considerable generality, new results are expected to apply in a much broader class of cases [Binford 82b]. Another aspect is locating regularities in images without relating them to surface interpretations [Lowe 82]. The principle that the world perceived should be independent of the observer leads to scale invariance in addition to position and rotation invariance. This implies that grouping operations should be performed at all positions, all orientations if directional, and all scales. Structures can occur at multiple levels; in the ocean, for example, the water surface is described by ripples and eddies superimposed on waves at one scale, described as flat at a larger scale, and circular at the scale of the earth. The principle that allowable computational complexity is limited implies that only the lowest complexity grouping operations can be carried out. In particular, complexity linear in the number of features implies grouping of each element with a constant number of other elements. Here we intend grouping by proximity and neighborhood relations. Diameter-limited grouping is not effective here; what is appropriate is complexity-limited grouping up to the limit of number of neighbors.

A measure is defined of the likelihood of random occurrence of constellations, using non-parametric statistics, without a priori knowledge of distributions. An example is described of determining linear features in dot patterns.

REFERENCES

[Arnold 77] Arnold,R.D.; *"Spatial Understanding"* Proc Image Understanding Workshop, April 1977.

[Arnold 80] Arnold, R.D., Binford, T.O.; *"Geometric Constraints in Stereo Vision"* Proc SPIE Meeting, San Diego, Cal, July 1980.

[Baker 82] Harlyn Baker; *"A system for automated stereo mapping"*; These proceedings, 1982.

[Binford 81] Binford, T. O.; *"Inferring Surfaces from Images"*; Artificial Intelligence Journal August, 1981.

[Binford 82a] T.O.Binford, B.Katz, M.R.Lowry, P.H.Winston; Forthcoming Joint Stanford-MIT AI memo, 1982.

[Binford 82b] T.O.Binford; *"Inferring Objects: Figure-Ground Mechanisms"*; in preparation.

[Binford 82c] T.O.Binford, ed; in preparation.

[Binford 82d] T.O.Binford; *"Survey of Model-Based Image Analysis Systems"*; International Journal of Robotics Research, 1982.

[Brooks 80] Brooks, R.A.; *"Goal-Directed Edge Linking and Ribbon Finding"*; Proc Image Understanding Workshop, Palo Alto, Calif, Apr 1979.

[Brooks 81] Brooks, R.A.; *"Symbolic Reasoning among 3-D Models and 2-D Images"*; Artificial Intelligence Journal, August, 1981.

[Lowe 81] Lowe, D. and Binford, T.O.; *"The Interpretation of Three-Dimensional Structure from Image Curves"*; Proc Int Joint Conf on AI, Aug 1981.

[Lowe 82] David Lowe and T.O.Binford; *"Segmentation and Aggregation: An Approach to Figure-Ground Phenomena"*; These proceedings, 1982.

[Lowry 82a] Michael Lowry; *"Reasoning between structure and function"*; These proceedings, 1982.

[Lowry 82b] Michael Lowry and Allan Miller; *"Analysis of low-level computer vision algorithms for implementation on a VLSI processor array"*; These proceedings, 1982.

[Malik 82] J.M. Malik and T.O. Binford; *"Representation of time and sequences of events"*; These proceedings, 1982.

[Marimont 82] D. Marimont; *"Segmentation in Acronym"*; These proceedings, 1982.

AD-P000207

MIT PROGRESS IN UNDERSTANDING IMAGES

Michael Brady, and the staff

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

In this series of Image Understanding Workshop Proceedings, we have stressed the issue of representation. In particular, we have described the development by Horn and his colleagues of the reflectance map, *the* albedo image, *and the* Gaussian image, *and we have described the work of the group founded by Marr using the* primal sketch, *the* $2\frac{1}{2}$ D sketch, *and axis based* 3 -D models.

In the April, 1981 Proceedings, we reviewed work on computing shape from shading and occluding boundaries, including a local parallel algorithm for doing this due to Ikeuchi and Horn [1981]; the detection and perception of motion by computing the optical flow and directional selectivity of zero crossings; the interpolation of curves and surfaces; the real-time convolution of images with a difference-of-Gaussians (DOG) operator; and progress toward computing the full primal sketch.

Here we review work on stereo *to facilitate the computation of depth information and visible surface characteristics, the* detection and interpretation of motion, *the* interpolation and description of visible surfaces, *the* description of two- and three-dimensional shapes, real-time convolution, and shape from shading.

1. Stereo

We have previously described the theory and implementation of Marr and Poggio's theory of human stereo [Marr and Poggio, 1979; Grimson 1981a, 1981b]. The left and right images are first convolved with a number of DOG masks and the zero crossings are then matched using a coarse-to-fine strategy We have made a number of modifications to the published algorithm, and are testing the current version on natural aerial images. The published algorithm incorporates a continuity constraint that is applied over horizontal slices of (rectified) images. Individual horizontal slices are treated separately. Taking note of the work of Mayhew and Frisby [1981] and Baker and Binford [1981], we have developed a continuity constraint that checks for consistency along zero-crossing contours that typically correspond to a single physical edge. We have also investigated the sensitivity of the algorithm to vertical disparity and other image distortions.

We have discovered further Psychophysical support for the stereo theory. It was shown by Julesz that random-line stereograms with tiny breaks (in the vernier acuity range) can be successfully fused. This has been interpreted as suggesting that the interpolation process underlying hyperacuity is parallel with, and preliminary to, stereo matching. Nishihara and Poggio [1982] have demonstrated that vernier cues are not needed to perform stereo matching since zero-crossings contain sufficient information.

The current version of the stereo algorithm, including the Marr-Hildreth theory of edge detection, is being tested on a variety of aerial images. Two of these, supplied by the Defense Mapping Agency and the University of British Columbia (UBC) are of natural terrain. A third stereo pair, also supplied by UBC, is of a building complex and features a deep quadrangle surrounded by high rise offices. The fourth pair, supplied by Boeing Corporation, is of a complex highway intersection. Preliminary results of running the stereo algorithm on these images are encouraging, and suggest that the algorithm is robust and does not require tuning for different classes of stereo pairs.

Recently, we have begun experimentation with a novel, fast stereo algorithm due to Nishihara that gives limited information. It may have important practical applications. Suppose that the two cameras are focused on a plane (called the focus plane). The matcher rapidly determines whether a zero-crossing is in front of, on, or behind the focus plane. Implemented in LISP machine microcode, the algorithm takes about 2 seconds to process a 1000 pixel squared image.

We have investigated the mathematical relationship between the Marr-Poggio theory of stereo and Horn's work on shape from shading. Grimson [forthcoming] has shown that if the reflectance map [Horn and Sjoberg 1979] is known, then given a pair of stereo matched depth contours it is possible to determine the surface normal along the depth contour. The proof suggests a technique for finding surface normals that is essentially analogous to photometric stereo, pioneered by Horn, Woodham, and Silver [1978]. Conversely, it is possible in principle to determine certain visible surface characteristics from stereo information.

Suppose that the reflectance map is of the form

$$R(\mathbf{n}) = \rho\left[(1-\alpha)(\mathbf{n}-\mathbf{s}) + \alpha(\mathbf{n}-\mathbf{h})^k\right],$$

where ρ is the albedo, α determines the convex combination of the specular and matte components of the reflectance, and k is the degree of specularity. Assuming k is known, it is possible to determine ρ and α. It is also possible in principle to estimate k over an area of the image. With the separation of human eyes, the technique is most effective at a distance of about one meter. However, the technique may find application to wide angle stereo.

2. The detection and perception of motion

The last Proceedings contains a description of the algorithm invented by Horn and Schunck [1981] for computing optical flow. Optical flow is the distribution of velocities of apparent movement caused by smoothly changing brightness patterns. The algorithm works well on synthetic images, especially when there are no depth boundaries in the scene. It also appears to give reasonable results when there are depth boundaries, though the errors in the flow become significant at the boundary. Schunck is continuing to develop the algorithm, to make it more robust, less sensitive to noise, and applicable to a wide variety of natural images.

Recently, Bruss and Horn [1981] (see this volume) have proposed a method for interpreting the optical flow. The technique is applicable to automatic passive navigation. More precisely, a method is proposed for determining the motion of a body relative to a fixed environment using the changing image seen by a camera attached to the body. The optical flow in the image plane is the input, while the instantaneous rotation and translation of the body are the output. If optical flow could be determined precisely, it would only have to be known at a few places to compute the parameters of the motion. In practice, however, the measured optical flow is rather inaccurate. It is therefore advantageous to consider methods which use as much of the available information as possible. Bruss and Horn employ a least squares approach which minimizes an appropriate measure of the discrepancy between the measured flow and that predicted from the computed motion parameters. Several different error norms have been investigated. In the general case, the algorithm leads to a system of nonlinear equations from which the motion parameters may be computed numerically. In the special cases of pure translatory or rotational motion, use of the appropriate norm yields a system of equations that are solvable in closed form.

In other work on motion detection, Hildreth and Ullman [forthcoming] have developed a technique that combines features of Horn and Schunck's work on optical flow with Marr and Ullman's [1981] work on directional selectivity, in which motion is detected from the temporal changes in zero-crossings. Marr and Ullman suggested that the initial computation of motion takes place at the location of features in an image, in particular at zero-crossings. Due to the local nature of the measurement of motion along zero-crossing contours, the motion is only determined perpendicular to the zero-crossing contour. A subsequent processing stage is required to integrate the local measurements in order to compute the component tangential to the zero-crossing contour.

Following Horn and Schunck's work on optical flow, the integration of local motion measurements is based upon a definition of smoothness or measure of local variation in motion. Hildreth and Ullman have explored several definitions of smoothness. It turns out that the optimal measure of local variation is the integral of the absolute value of $\partial v/\partial s$, where s denotes arclength and v is the velocity field. It can be shown that there is a unique velocity field that satisfies the initial motion measurements and minimizes the measure of variation along a zero-crossing contour. We are now proceeding to implement the method.

The spatial resolution of an image is limited by the sampling density of the photosensitive elements in the sensor and by noise. Image motion introduces the additional problem of temporal resolution. The limiting factors are the frame rate and the integration time of the photosensitive elements. This is of little consequence for a stationary scene, but for moving targets, it poses the problem of motion smear.

The problem of high spatio-temporal resolution can be overcome partly by using better sensors with larger arrays and higher frame rates. There are, however, technological and physical limits to the spatio-temporal resolution that can be achieved in this manner, since increasing the spatial and temporal sampling rate reduces the number of photons per sensor element per cycle. The performance of a given sensor can be improved by appropriate spatio-temporal interpolation schemes. Using such interpolation processes, the human visual system achieves an extremely high spatio-temporal resolution compared to the sampling density of the photoreceptors and their integration time.

There are various methods for reconstructing the original signal at high resolution by interpolating values measured at widely spaced intervals. The best known approach to this problem is based on the Shannon sampling theorem and on its various extensions. Although it is usually formulated for one-dimensional signals, its extension to two dimensional time-varying images is straightforward.

For static images, interpolation of this type can provide a resolution much higher than the original sampling grid. Since, in our framework, the position of zero-crossings is important, Hildreth and Poggio have examined the problem of interpolating the values of the DOG convolution in order to obtain precisely the location of zero-crossings. Analytical arguments, supported by computer experiments, have shown that the position of a zero-crossing can be interpolated precisely in terms of very simple interpolation functions, even in fact by linear interpolation.

For time-varying images the situation is more complex. In the classical scheme, interpolation in space and time is performed independently, since the temporal dependence of the input is not constrained in any way.

More effective interpolation schemes are feasible if general constraints about the nature of the visual input are incorporated directly into the computation. The key observation here is that the temporal dependence of the visual input is usually due to the movement of rigid objects, and such motion usually has a nearly constant velocity for the short time and distance over which the interpolation process operates. Under this constant-velocity assumption, Poggio has shown that the spatio-temporal Fourier spectrum of a moving image is strongly constrained leading to a new form of the sampling theorem:

Interpolation schemes based on the constant velocity assumption exploit the equivalence of the time and space variable. From the point of view of filtering, this means that spatial and temporal interpolation cannot be performed independently. Interpolation algorithms based on this approach could achieve high spatio-temporal resolution for objects in motion, as long as the constant velocity assumption is not grossly incorrect, despite low spatial and temporal sampling rates. High positional acuity for the image features, although desirable for tracking moving targets, is not the only goal of spatio-temporal interpolation. A filter that correctly interpolates the sampled image automatically avoids any defect in the representation of the image since it reconstructs the original input. It avoids in particular motion smear, and it fills in eventual gaps either in space or time, wherever or whenever the sampled input is missing.

3. The interpolation and representation of surfaces

In our last progress report, we described our work on interpolating curves and surfaces. Interpolation is an important problem for vision, since several visual processes, notably stereo and structure from motion, only specify depth and orientation at a discrete subset of the points in an image. The points at which they are specified are typically those where the irradiance changes abruptly. In the Marr-Hildreth theory of edge detection, such points coincide with the zero-crossings of a DOG operator applied to the image. Human perception, however, is of complete, piecewise smooth, surfaces, and such complete surface information is important for most applications of vision. Since mathematically the class of surfaces which could pass through the known boundary points provided by stereo, for example, is infinite and contains widely varying surfaces, the visual system must incorporate some additional constraints in order to compute the complete surface.

Using the image irradiance equation formulated by Horn [1978], Grimson [1982] has derived a ***surface consistency constraint***, informally known as "no news is good news". The constraint implies that the surface must agree with the information from stereo or motion correspondence, and *not* vary radically between these points. An explicit form of the surface consistency constraint has been derived, by relating the probability of a zero-crossing in a region of the image to the variation in the local surface orientation of the surface, provided that the surface albedo and the illumination are roughly constant.

A second idea is that, in the absence of contrary evidence, the visual system constructs the *most conservative* curve or surface consistent with the given sparse data. This is made precise using the calculus of variations. A crucial aspect of the variational formulation is the choice of performance index to minimize. Grimson [1981b] argued that the performance index should be a seminorm, and suggested the quadratic variation $f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2$. Brady and Horn [1981] (see also this volume) have noted that any quadratic form in f_{xx}, f_{xy}, and f_{yy} is a seminorm, and so is a plausible performance index. They have shown that the quadratic forms that are, in addition, rotationally invariant form a vector space, which has the square Laplacian and the quadratic variation as a basis. Since the quadratic variation has the smaller null space, it offers the tighter constraint, and is to be preferred. Brady and Grimson [1981] have suggested that surface perception is the basis for the the perception of subjective contours.

Brady and Horn [1981] suggest that surface interpolation can be posed in terms of a physical model, namely as the variational problem describing the constrained equilibrium state of a thin flexible plate. The variational problem and the physical model have been developed by Terzopoulos [1982]. After formulating surface interpolation as an energy minimizing problem over an appropriate Sobolev space, the problem is discretized and approached via the finite element method. In essence, the variational problem is transformed into a large set of linear algebraic equations whose solution is computable by local-support, cooperative, parallel processors.

It has been suggested that visual processes such as edge detection and stereo provide information at a number of distinct scales, spanning a range of resolutions. To exploit the information available at each level of resolution, a hierarchy of discrete problems is formulated and a highly efficient multi-level algorithm, involving both intra-level relaxation processes and bi-directional, inter-level, local, interpolation processes, is applied simultaneously to discover the solution. Intra-level relaxation smooths out high frequency variations, while inter-level interpolation tends to damp out low frequency variations, greatly speeding the overall process. The resulting process is extremely efficient, even on a serial computer, though it is better suited to an array of parallel processors. Current work is concentrating on the isolation of surface discontinuities and the integration of the results of several visual processes such as shape from shading, structure from motion, and stereo, toward a deeper understanding of the structure of the $2\frac{1}{2}$-D sketch.

Once complete surfaces have been interpolated from sparse data, it is necessary to describe them, for example to facilitate recognition or inspection. Horn [1982] has suggested the Gaussian image as a representation of surface shape. Local surface normals are brought together at an origin. Parallel normals are represented by a vector in their common direction. The magnitude of the vector is proportional to the number of normals sharing that orientation. A theorem of Minkowski shows that the representation is faithful for convex objects, in the sense that the original surface can be recovered from the Gaussian image. The scheme is currently being implemented.

In other work on the same problem, Brady [1982c] has proposed a representation of visible surfaces based on *curvature patches*, which are surface patches similar to those used in computer-aided design (CAD), but which differ in two respects. First, the webbing, or surface parameterisation, is required to consist of (a suitable tesselation of) lines of curvature on the surface. Second, the blending function based on the approach to surface interpolation developed by Grimson, Horn, Brady, and Terzopoulos. It is shown that the representation is complete, and that it has a number of advantages over conventional CAD representations, such as bicubic splines or Bezier surfaces. Surface intersections are represented in a way that generalizes techniques associated with line drawing analysis, and is related to the work of Binford [1981]

4. Shape description

The description of two- and three-dimensional shape is crucial for recognition. Brady [1982a, 1982b] has developed a representation of two-dimensional shapes that combines certain features of two-dimensional projections of generalized cylinders [Nevatia and Binford 1977, Brooks 1981] and the symmetric axis transform (SAT) [Blum and Nagel 1978]. The representation has four components. First, local symmetry is defined in a way that differs from that implicit in the SAT. Second, axes that are smooth loci of local symmetries are computed. Third, axes whose region of support is wholly subsumed by some other axis are deleted. The resulting *smoothed local symmetries* are given a parametric description called a frame. Finally, consistent frames from adjacent pieces of shape are propagated to form an overall shape description.

A pilot implementation of smoothed local symmetries has been constructed. It embodies an efficient algorithm, based on the mean value theorem, for determining the points at which a line entering the shape at a given orientation to the tangent emerges from the shape. The representation has been applied to determine where to choose grasp points on a lamina for a two-fingered robot hand. Currently the representation is being implemented for smoothly curved shapes and for shapes with straight edges.

The implemented system extracts the bounding contours from images using DOG filters. Recently, however, an improved line finder has been developed by Canny [forthcoming]. Canny has shown that the first derivative of a Gaussian is the filter that optimizes the product of a measure of the detectability of edges, and a measure of their localization. The operator is directional and operates at a number of scales. An operator that finds lines by nonmaximal suppression has also been developed.

5. Reflectance techniques

Reflectance techniques can be applied to recover descriptions of ground cover from remotely sensed images. Working with Horn [Horn and Sjoberg 1980], Sjoberg [1982] has shown that certain strong, though reasonable, assumptions permit the use of a simple parametric image forming equation appropriate to satellite sensing. Topographic effects can be determined with the aid of a digital terrain model for the area imaged. Atmospheric effects can be estimated, partly from the image itself and partly by tuning the model parameters and subjectively evaluating the resulting synthetic albedo images. The subjective criteria include: no shading artifacts due to the topography, there should be a close match between the sunlit and shadowed areas straddling a cast shadow boundary, and the dynamic range of the computed albedo is limited. This kind of tuning is necessary if, as is often the case, no additional information about the scene or the atmosphere can be obtained.

6. Learning physical descriptions from functional definitions, examples, and precedents

Working with Binford and Lowry of Stanford University, Winston and Katz have developed a theory of learning that explains how physical descriptions for recognition can be generated using functional definitions, particular examples, and precedent knowledge. The work synthesises two sets of ideas: ideas about learning from precedents and exercises developed by Winston at MIT, and ideas about physical description developed in the ACRONYM system at Stanford.

7. References

Baker H. Harlyn, and Binford T. O. "Depth from edge and intensity based stereo," *Int. Jt. Conf. Artif. Intell.* **6** (1981), .

Binford T. O. "Inferring surfaces from images," *Artificial Intelligence* **17** (1981), 205-245.

Blum, Harry, and Nagel, Roger N. "Shape description using weighted symmetric axis features," *Pattern Recognition* **10** (1978), 167-180.

Brady, Michael. "Parts description and acquisition using vision," *Proc. SPIE, Washington D.C.,* (1982a), .

Brady, Michael. "Smoothed local symmetries and local frame propagation," *Proc. Patt. Rec. and Im. Proc., Las Vegas, June* (1982b), .

Brady, Michael. "Criteria for representations of shape," *Human and Machine Vision* eds. Rosenfeld A., and Beck J., 1982c.

Brady, Michael and Grimson, W. E. L. "The perception of subjective surfaces," MIT, AI Memo 666, 1981.

Brady Michael, and Horn B. K. P. "Rotationally symmetric operators for surface interpolation," MIT, AI Memo 654, 1981.

Brooks R. "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intelligence* 17 (1981), 285-349.

Bruss A., and Horn, B. K. P. "Passive Navigation," MIT, AI Memo 662, 1981.

Mayhew J. and Frisby J. P. "Psychophysical and Computational studies toward a theory of human stereopsis," *Artificial Intelligence* 17 (1981), 349-387.

Grimson, W. E. L. "A computer implementation of a theory of human stereo vision," *Phil. Trans. Roy. Soc. Lond. B* **292** (1981a), 217-253.

Grimson, W. E. L. *From images to surfaces: a computational study of the human early visual system* , MIT Press, Cambridge, 1981b.

Grimson, W. E. L "The implicit constraints of the primal sketch," MIT, AI Memo 663, 1982.

Horn B. K. P. "Understanding image intensities," *Artificial Intelligence* 8 (1977), 201-231.

Horn B. K. P. "Hill shading and the reflecta[illegible] map," *Proc. IEEE* 19 (1981), 14-47.

Horn B. K. P. "Sequins and Quills - Representations for Surface Topography," *Representation of 3-Dimensional Objects* ed. Bajcsy R., Springer Verlag, 1982.

Horn B. K. P. and Schunck B. G. "Determining optical flow," *Artificial Intelligence* 17 (1981), .

Horn B. K. P. and Sjoberg Robert W. "Calculating the reflectance map," *Appl. Optics* 18 (1979), 1770-1779.

Horn B. K. P. and Sjoberg Robert W. "Atmospheric modelling for the generation of albedo images," *Proceedings of the Image Understanding Workshop* ed. Baumann Lee, Science Applications, 1980.

Horn B. K. P., Woodham R. J., and Silver W. M. "Determining shape and reflectance using multiple images," MIT, AI Memo 490, 1978.

Marr D. and Hildreth E. "Theory of edge detection," *Proc. R. Soc. Lond. B* **207** (1980), 187-217.

Marr D. and Poggio T. "A theory of human stereo vision," *Proc. R. Soc. Lond. B* **204** (1979), 301-328.

Marr D. and Ullman S. "Directional selectivity and its use in early visual processing," *Proc. R. Soc. Lond. B* **211** (1981), 151-180.

Nevatia R. and Binford T. O. "Description and Recognition of curved objects," *Artificial Intelligence* 8 (1977), 77-98.

Nishihara, H. K., and Poggio, T. "Hidden cues in random line stereograms," MIT, AI Working Paper 230, 1982.

Sjoberg, R. W., and Horn, B. K. P. "Atmospheric effects in satellite imaging of mountainous terrain," MIT, 1982.

Terzopoulos, D. "Multi-level reconstruction of visual surfaces," MIT, AI Memo 671, 1982.

AD-P000 10 8

THE SRI IMAGE UNDERSTANDING PROGRAM*

M. A. Fischler (Principal Investigator)

SRI International, Menlo Park, California 94025

ABSTRACT

Our principal objective in this research program is to obtain solutions to fundamental problems in computer vision that have broad military relevance, particularly in the areas of cartography and photo interpretation. Current research is directed towards developing automated high-performance techniques for stereocompilation, delineation of linear features, scene partitioning (and material identification), and image matching (and image to database correspondence).

In addition to our own research, we have designed and are implementing an integrated testbed system that incorporates the results of research produced throughout the image understanding community. This system will provide a coherent demonstration and evaluation of accomplishments of DARPA's image understanding program, thereby facilitating transfer of this technology to appropriate military organizations.

I INTRODUCTION

Research at SRI International under the DARPA Image Understanding Program was initiated to investigate ways in which diverse sources of knowledge might be brought to bear on the problem of analyzing and interpreting aerial images. An initial exploratory phase of research identified various means for exploiting stored knowledge in the processing of aerial photographs for such military applications as cartography, intelligence, weapon guidance, and targeting. A key concept is the use of a generalized digital map to guide the process of image analysis. The results of this earlier work were integrated into an interactive computer system called "Hawkeye" [1]. This system provides necessary basic facilities for a wide range of tasks in cartography and photo interpretation.

Research subsequently focused on development of a program capable of expert performance in a specific task domain--road monitoring. The primary objective of this work has been to build a computer system, called the Road Expert, that "understands" the nature of roads and road events. It is capable of performing such tasks as

* Finding roads in aerial imagery.
* Distinguishing vehicles on roads from shadows, signposts, road markings, etc.
* Comparing multiple images recorded at different times with symbolic information pertaining to the same road segment, and deciding whether significant changes have occurred.

The general approach, and technical details of the Road Expert's components are contained in References [2-8]. We have integrated these separate components into a coherent system that facilitates testing and evaluation, and have transferred this system to the DARPA/DMA Testbed.

A parallel research program (described in Reference [7]), jointly supported by DARPA and NSF, has complemented the above investigations by focusing on fundamental computational principles that underlie the early stages of visual processing in both man and machine.

At present we are involved in two major efforts. The first is in support of a joint DARPA/DMA program to provide a framework for demonstrating and evaluating the applicability of image understanding research (from throughout the entire IU community) to military problems in general, and to the problems of automated cartography, in particular. Our plans and progress in this effort (DARPA/DMA Testbed) are described in a separate paper in these proceedings (Reference [9]).

The goal of our second major effort is to carry out a broad program of machine vision research -- specifically in the areas of three-dimensional terrain understanding, linear-feature analysis, image partitioning, and image description and matching. This research program has been centered on the concept that image interpretation, except in the simplest situations, involves a form of reasoning ("perceptual reasoning") characterized by the need to integrate information from multiple sources that are typically incommensurate and often erroneous or in conflict.

We have developed a number of new techniques, and even complete paradigms, for effecting the knowledge integration task. These new techniques have been incorporated in the more focused efforts, discussed below, which address significant problems in scene analysis.

----

* The research described in this paper is based on work performed under Advanced Research Projects Agency Contract No. MDA903-79-C-0588.

II RESEARCH PROGRESS AND ACCOMPLISHMENTS

A. Three-Dimensional Compilation and Interpretation

The problem of stereo reconstruction is almost synonymous with the problem of machine vision: use of imaged data to (geometrically) model a sensed scene. We are constructing an integrated system for stereo compilation to satisfy two goals: first, to provide a framework for the development and evaluation of a number of our own efforts, as well as those of the IU community, in scene modeling; second, to advance the state of the art of a critical task domain in cartography.

A key concept in our approach is the use of global physical and semantic constraints (e.g., the sun's location, vanishing points, edge detection and classification, skyline delineation, etc.) to provide a means of resolving local ambiguities that frustrate conventional stereo matching techniques in mapping cultural or urban scenes. Such scenes contain featureless areas and large numbers of occlusion edges or, alternatively, are represented by widely separated or oblique views. We are taking special measures to make our system modular so that critical components, embodying work done at other IU centers and present in the testbed, can be freely substituted for existing modules. References 10-14 present a more complete discussion of our work in this research area.

B. Detection, Delineation, and Interpretation of Linear Features in Aerial Imagery

We have developed a system, called the "Road Expert," that can precisely delineate roads in both high- and low-resolution aerial imagery, and can classify the visible objects that fall within the road boundaries [2-8]. A demonstration version of the Road Expert has been installed on the DARPA/DMA testbed. We have investigated extensions of this work to the problem of delineating more general types of linear structures and are presently developing techniques that will enable the system to adjust its own parameters to optimize performance over a variety of viewing conditions and terrain types without the need for operator intervention. Our ultimate goal is a high-performance, completely autonomous system for linear delineation.

C. Image Matching and Image-to-Database Correspondence

We have developed a new paradigm, called Random Sample Consensus (RANSAC), for fitting a model to data containing a significant percentage of gross errors, and have applied this paradigm to the solution of the matching/correspondence problem [15]. A RANSAC-based camera model solver has been developed and installed on the testbed. We expect that RANSAC will be equally applicable to a wide range of other model-based interpretation tasks, and are presently investigating its use in techniques for recognizing and labeling known two- and three-dimensional scene features, even though seen under unusual viewing or illumination conditions, and even when the objects are partially occluded.

D. Image Partitioning, Intensity Modeling, and Material Identification

Our goal in this effort is to develop techniques for partitioning and modeling the material composition of a scene from available imagery. In order to recover information about actual surface reflectances and physical composition, the problem of intensity modeling must be addressed. We have devised methods for deriving absolute scene-intensity information in the absence of calibration data (such as a step wedge exposed on the image), based on knowing the identity of the material composition of the surfaces at a few locations in the image -- a necessary capability for partitioning the image into labeled regions of a given material type (see Reference [11]).

ACKNOLWEDGEMENTS

Contributors to the SRI Image Understanding Program include the following staff members: S. Barnard, R. C. Bolles, A. Farley, K. Laws, M. Lowry, M. A. Fischler, A. J. Hanson, D. L. Kashtan, A. P. Pentland, G. Smith, and H. C. Wolf.

REFERENCES

1. H. G. Barrow et al., "Interactive Aids for Cartography and Photo Interpretation: Progress Report," in Proceedings of the Image Understanding Workshop, Palo Alto, California, pp. 111-127 (October 1977).

2. M. A. Fischler et al., "Interactive Aids for Cartography and Photo Interpretation," Semiannual Technical Report, SRI Project 5300, SRI International, Menlo Park, California (October 1978 and May 1979).

3. L. Quam, "Road Tracking and Anomaly Detection," in Proceedings of the Image Understanding Workshop, Boston, Massachusetts, pp. 51-55 (May 1978).

4. R. C. Bolles et al., "The SRI Road Expert: Image-to-Database Correspondence," in Proceedings of the Image Understanding Workshop, Palo Alto, California, pp. 163-174 (November 1978).

5. G. J. Agin, "Knowledge-Based Detection and Classification of Vehicles and Other Objects in Aerial Road Images," in Proceedings of the Image Understanding Workshop, Los Angeles, California, pp. 66-71 (April 1979).

6. R. C. Bolles et al., "Automatic Determination of Image-to-Database Correspondence," in Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan, pp. 73-78 (1979).

7. M. A. Fischler, "The SRI Image Understanding Program," in Proceedings of the Image Understanding Workshop, pp. 152-155 (November 1979).

8. M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf, "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," Computer Graphics and Image Processing, Vol. 15, No. 3, pp. 201-223 (March 1981).

9. A. Hanson, "The DARPA/DMA Image Understanding Testbed," (in these proceedings).

10. S. T. Barnard and M. A. Fischler, "Computational Stereo," ACM Surveys (1982, in press).

11. M. A. Fischler et al., "Modeling and Using Physical Constraints in Scene Analysis," (in these proceedings).

12. S. T. Barnard, "Methods for Interpreting Perspective Images," (in these proceedings).

13. G. Smith, "The Recovery of Surface Orientation from Image Irradiance," (in these proceedings).

14. A. P. Pentland, "Depth of Scene from Depth of Field," (in these proceedings).

15. M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," CACM, Vol. 24, No. 6, pp. 381-395 (June 1981).

SECTION II

TECHNICAL PAPERS
PRESENTED

7000 109

IMAGE UNDERSTANDING APPLICATION PROJECT: STATUS REPORT

B.L. Bullock, G.R. Edwards, D.M. Keirsey, D.Y.Tseng, F.M. Vilnrotter

HUGHES RESEARCH LABORATORIES
Malibu, California 90265

and

D.H. Close, J.F. Bogdanowicz, H.A. Parks, D.R. Partridge, E.P. Preyss

ELECTRO OPTICAL AND DATA SYSTEMS GROUP
HUGHES AIRCRAFT COMPANY
El Segundo, California

ABSTRACT

This paper reviews the current status of an ongoing program to demonstrate the application of DARPA Image Understanding research to a photo interpretation system using real imagery. The program is based on the ACRONYM vision system, developed by Rod Brooks and Tom Binford at Stanford University on the DARPA Image Understanding Project. This system was chosen for its sophistication and potential for extension. ACRONYM is running in FRANZLISP on a VAX with the DARPA IU Testbed environment. Current work on the project involves extending ACRONYM to meet the specific application requirements. These extensions include improving the robustness of the low level description module, broadening the class of information generated by the prediction system (including the ability to predict shadows), and adding a script driven situation assessment module. This latter module will provide the capability to understand a series of interpretation results.

1.0 INTRODUCTION

Research in Image Understanding (IU) has been conducted for the last 20 years at several university and industrial laboratories. In recent years, the major focus of this research has been the DARPA Image Understanding Project. Research on both vision system components and complete vision systems has been reported in the proceedings of the Image Understanding Workshops. The history of these developments is well known in the IU community. An IU testbed [7], [8] has been established to facilitate use of this research in applications. Hughes has been performing on a program, supported by DARPA and ONR, to apply research results from the DARPA Image Understanding Project. The objective of this program is to automate parts of an operational photointerpretation task. The approach is to construct a system from selected components and demonstrate its capabilities on real imagery using a general purpose computer.

The system we are constructing will search digitized images for instances of interesting objects. The system is both top-down (prediction-driven) and bottom-up (data-driven). Both camera and illumination models are included. Image and shadow shape predictions are being generated symbolically from 3-D object models. A sophisticated symbolic matching process is being used to interpret line finder results. Objects observed in a sequence of images are related to expected event sequences via a situation assessment module.

This paper describes the progress to date and status of the Hughes IU project. Section 2 summarizes progress, including the IU component selection, identification of necessary extensions, and work to carry out these extensions. Section 3 describes work in progress. Section 4 gives a prognosis and plans. The ACRONYM system, which is the basis of our development effort, is described briefly in the appendix.

2.0 PROGRESS TO DATE

The IU applications project began with an extensive evaluation of available IU software in terms of the application requirements. Both IU testbed software at SRI and software at other laboratories were surveyed. Both vision components and vision systems were evaluated. The study was documented in a report, "IU Algorithm Report," March, 1982. The ACRONYM system [3], [4], [5], [6] was chosen as the basis for the demonstration. Acronym's overall flexible implementation provided an excellent basis for the extensions required.

ACRONYM was developed by Rod Brooks and Tom Binford at Stanford University. Its vision knowledge is expressed in rules. Its functional capabilities include an interpretation system utilizing a 3-D model hierarchy and a powerful constraint system to guide the matching process. It was implemented in MACLISP, ran on a PDP-10, and was demonstrated in the domain of interpretation of aerial photographs of wide-bodied jet aircraft. The modular implementation is based on a core of record structures and control constructs. Other features of ACRONYM are summarized in the appendix.

The application objectives require a system with the functions indicated in Fig. 2. Following the image/model matching process, observed objects are related to symbolic history files in order to track a sequence of activities. This step, labeled Situation Assessment in the figure, is an addition to ACRONYM, which provides the image/model matching activities.

Analysis of the 600 pages of ACRONYM source code showed that several extensions to the ACRONYM system are needed for the desired application. These extensions fall into five categories: (1) more complex control structure, (2) prediction of shadows, (3) improved extraction and matching of line segments, (4) means for relating observations from a sequence of images, and (5) further application-level development tools. These five extension areas are in various stages of detailed design and implementation. The status of each area is discussed in the following section.

The ACRONYM system is running in FRANZLISP on a VAX 11/780 at Hughes. FRANZLISP itself is supported on the VAX by the EUNICE environment, provided by the IU Testbed. The computing environment is summarized in Fig. 1.

3.0 WORK IN PROGRESS

The remainder of this section reports the progress and status of the five extension areas.

3.1 SEQUENCING AND CONTROL

Coordination of system activities is accomplished by a set of sequencing and control rules. Their task is to relate model-driven (top-down) predictions and data-driven (bottom-up) information derived from the image. The topmost level of the sequencing and control rule hierarchy is shown in Fig. 3. First, the current image is interpreted. Next, these results are related to historical data. Finally, results from the image interpretation and situation assessment steps are reported.

Sequencing and control of the image interpretation process consists of two major steps. First, the rule (SC-SREGISTRATION) matches the image scene to a geographic model.

When an adequate location match is obtained, the second image interpretation step identifies pre-selected regions of interest in image coordinates. These regions are formed into a list of windows by rule (SC-WINDOW-SELECT). The system then iterates through this list one or more times, seeking to locate and identify object instances. A lower-level tier of control rules directs the search within each window and uses the system object/model hierarchy to identify objects. These rules issue as many calls to the vision system as needed, each seeking to match a specific object model from the model hierarchy to extracted image features.

Within the vision system, separate rule subsets control the prediction, ribbon-finding, and model matching steps. The prediction step calculates observable surface shapes and their relationships, for both objects and their shadows. Line and ribbon finding, described in Section 3.3, utilizes heuristic rules for line finder parameter selection. The model matching process matches prediction graph nodes and then prediction graph arcs.

Control within the situation assessment and report generation steps is sequential. The five situation assessment steps are script selection, event prediction, script verification, script updating, and script inferencing. Report generation is initiated by the rule (SC-REPORT-GENERATION).

The design of the high-level structure of Fig. 3 has been completed. In addition the overall sequencing and control of the vision system is complete. This consists of the ACRONYM control structure together with modifications for shadow prediction (see Section 3.2) and a more sophisticated shape extraction design (see Section 3.3). The next step is writing and checking out the control rules.

3.2 SHADOW EXPLOITATION

A shadow prediction capability is being added to Acronym in order to exploit all observed image features. The approach taken is to analyze Acronym's prediction process and define extensions needed for implementing the shadow understanding capability.

Objects in Acronym are modeled as combinations of generalized cones [1], [10]. Each cone gives rise to a set of planar contours which correspond to faces of a cone or, in the case of a curved surface, its projection onto a plane. Each cone has its own coordinate system.

Acronym's prediction process has three major functions. First, given a specific object cone, a camera, and their locations and orientations in a world coordinate system, Acronym produces a simplified transformation of the cone into the camera's coordinate system. The rotation expression of this transformation is a product of rotations. Acronym predicts how these individual rotations distort the shapes of contours which are expected to appear in the image. Secondly, Acronym predicts spatial relationships between the subparts of an object, i.e. between the spines of the cones comprising the object. Finally, Acronym prediction provides direction for the low-level descriptive process, i.e. the ribbon finder.

Acronym's ability to predict the above image features is mainly limited by the camera's orientation with respect to the object. With help from Rodney Brooks, additions and modifications to the system are being made to allow an arbitrary camera orientation. Also, Acronym handles occlusion only in the case where the faces of two subparts of an object are hard up against each other. The ability to handle other cases of occlusion depends heavily on combining spatial information obtained from predictions of different subparts of an object and from different objects.

The Acronym prediction system is being extended to include shadow prediction. Given an illumination direction, the system will first predict whether or not a planar surface is illuminated. (See Fig. 4(a).) In the case of a curved surface, the location of the illumination boundary is predicted. (See Fig. 4(b).) In both cases the shadows cast by the object onto a background plane are predicted. Then, the dimensions of both the shadowed and illuminated contours of the object are predicted, as well as the dimensions of the shadow contours on the background plane. Next, the spatial relationships between object subparts and their shadows are predicted (Fig. 5). Finally, distortion of shadow contours due to individual rotations is predicted (Fig. 6).

The extensions required for shadow prediction are in a state of detailed design. This capability will be implemented over the next several months.

3.3 SHAPE EXTRACTION

To a great extent, the performance of a practical machine vision system is determined by its ability to reliably extract recognizable features from real-world imagery. In order to provide the ACRONYM system with this necessary element, a new design has been undertaken to extract two-dimensional shapes from images.

Much research has been directed toward the problem of linear feature extraction, resulting in several edge and line finding algorithms which perform quite well. However, much less progress has been made in techniques for extracting two-dimensional shapes.

There has been some recent work by Mulgaonkar, Shapiro, and Haralick [9] in which three-dimensional objects are modeled using blobs, plates, and sticks as object primitives. However, the goals of this work seem aimed toward the model representation and matching problems, rather than the specific problem of shape extraction. Their extraction of two-dimensional shapes relies upon simple point operators such as thresholding to segment objects, and representation of the extracted regions as near-convex polygons.

In general, region or blob feature extraction methods seem to be inadequate for application with the Acronym system. The power of these methods is evident in the relational matching that can be performed between these features and three dimensional models. However, features of this type are inherently weak in terms of the accuracy of the boundaries of the extracted objects, a key performance requirement for application with the Acronym system.

One approach to the two-dimensional shape extraction problem was exemplified by Brooks [2] in his Prototype Edge Mapping Module, which formed the shape extraction module for the original Acronym system. This approach begins by extracting linear edge segments with one of the existing line finding systems, such as the Nevatia-Babu linefinder [11]. Using this segment information, the shape extraction system attempts to trace boundaries of closed, convex two-dimensional contours. Segments are selected as boundary components based on connectivity or near-connectivity, and the tracing function is able to bridge colinear gaps and join nearby endpoints. These selected segments are then subjected to certain convexity tests to determine that they are tending toward formation of closed, convex contours. Finally, the traced contour is subjected to several tests of dimensionality and amount of enclosed area to determine if these parameters fall within some model-driven range of expectation. For each contour which survives these tests, a directionality histogram is computed, and this information is in turn used to describe the extracted shape as either an ellipse or ribbon. These ellipse and ribbon descriptions are directly analogous to the shapes predicted by Acronym's geometric reasoning system.

This method was able to perform adequately in some applications. However, the method has limitations for the desired application. One drawback stems from the fact that in many applications on real-world imagery, only a few boundaries of any object will be visible and detectable by the segmentation system. Occlusions of these bounding edges frequently occur due to other objects, shadows cast by nearby objects, and perhaps most seriously, self shadowing which is bound to occur in real imagery. In these cases, attempts to trace a complete, closed boundary seem doomed to failure. Another difficulty is that, once the contour has been traced, the approximation of a ribbon or ellipse based on directionality histograms results in relatively coarse shape approximations. While this drawback is not serious in some applications, it does limit the power of the vision system to identify objects based on detailed dimensional information. Finally, the trace and test approach makes very little use of information about the shapes which were actually predicted. While some information about the expected range of dimension and surface area is imbedded, it is again relatively coarse, resulting in the extraction of a great deal of clutter. Also, the shape extraction process makes no use of available information about the expected spatial connectivity of various predicted object subparts.

The design of a next-generation shape extraction system has been undertaken, building on Brooks' implementation. Several considerations have driven the design of this shape extraction module, with the goal of producing a system which would prove powerful enough to extract useful features yet provide robust performance in a wide range of applications.

The primary consideration lies in the concept of a top-down, model driven approach to shape extraction. Acronym predicts constrained instances of certain two-dimensional shapes from three-dimensional models, given some collateral information about the camera model. It is quite reasonable to provide these predicted shapes to the shape extraction system, and allow that system to search for specific instances of these predicted shapes.

By providing the shape extraction mechanism with this detailed information, specific instances of candidate shapes can drive selective searches for instances of specific segment-based features. For example, if a ribbon is predicted with a certain constrained range of width, length, and sweep characteristics, the shape extraction system can search for instances of segments which lie on parallel lines separated by a distance falling within the length range of the ribbon, and check that these segments fall within the proper spatial separation along these lines to potentially describe pieces of the end segments of the ribbon. Having identified some set of candidate segment pairs, search can be initiated for segments which would form parts of the side segments of the ribbon, with the possible separation and angular relationship determined by the width and sweep characteristics of the ribbon. A typical matching situation is indicated in Fig. 7.

Note that there is no need to find complete sides of a ribbon, nor is it necessary to find pieces of all four boundaries. Rather, candidates for matching may be selected based only on suggestive clues indicating the existence of an instance of a shape. Created along with this extracted shape would be some measure of the quality of this particular shape, in terms of some comparison of the perimeter of the predicted shape to the amount of boundary found, the number of sides where boundaries were supported, and the consistency of the spatial relations of the segments found.

Since there is no requirement to completely trace the boundaries of a contour, cases of partial occlusion by objects or shadows will not render the vision system incapable of matching these shapes. Therefore, the system should perform in a much more robust manner.

In addition to the model-driven approach to shape extraction, the new design also has knowledge of the spatial connectivity of various predicted shapes, especially shapes which represent subparts of a single modelled object. With this information, confidence in extracted shapes can be influenced by their connectivity to other shapes. For instance, there might be very low confidence in two shapes extracted from a small number of segments. However, if both of these shapes fall in the correct spatial pattern to describe the subparts they represent, there is much more confidence in the results.

Further, this knowledge of connectivity can be used to handle cases of occlusion of one subpart by a connected subpart. This type of reasoning is of particular advantage in situations where shapes projected from two subparts of an object overlap in such a way as to be seen in the imagery as a single, larger shape which can be identified by the extraction process.

The overall design of the new shape extraction module is shown in Fig. 8. The system is invoked from the executive rules which control the overall operation of the Acronym vision system. The initial rules select from an available set of shape-extraction heuristic rule sets, each of which may be expert in extracting certain types of features (ribbons, ellipses, linear features, etc.) or may specialize in shape extraction in certain imaging conditions (such as clear vs. hazy imagery, snowy conditions, etc.). Any one or any combination of these expert rule sets may be chosen by the selection rule set. The decision is guided by the shapes themselves, and also by information provided from outside the vision system (by the controlling system which calls the vision system as a sub-process), providing data on the imaging conditions.

As each heuristic rule set executes, the rules will need to gather specific types of edge-based features. The rules will be capable of parameterizing and invoking a line-finding process (such as the Nevatia-Babu system) or some other low-level processing or enhancement algorithms. The resulting edge features are then stored on a spatially indexed data base, along with more complex features derived from this edge information by the spatial indexing rules. These more complex features include parallel lines, line pair vertices, and colinear lines.

Having invoked the rules to extract edge data and create a library of features, the shape extraction rules access this data through requests to a set of rules which access the spatial feature data base. These rules in turn are influenced by spatial search restrictions which are imposed by the calling routine. These search restrictions may be used to guide searches for instances of objects near some known features or locations (such as searching for an airplane along a runway, rather than on top of a terminal building). The segment selection rules search the spatial feature library to provide any instances of the requested features within the restricted search area.

Having finally identified a particular shape, this shape is appended to a data structure called the Picture Graph, which contains all shapes to be submitted as match candidates. Knowledge of the existence of this particular shape is also appended to the spatial feature library, to provide cues for other shapes which may be predicted to be connected.

In all, this model-driven approach to two-dimensional shape extraction is expected to provide robust extraction performance over a wide range of application imagery.

The preliminary implementation of the design shown in Figure 8 is completed. For the desired application there will be considerable application-dependent knowledge embedded in the heuristic rule sets to control the details of shape extraction. This specific knowledge will be acquired by exercising the system with application imagery.

3.4 SITUATION ASSESSMENT

In the desired application, besides modeling the geometry of the objects in a scene, the overall situation must be modeled. That is, the actions of the objects have a predictable behavior and that behavior is modeled. More specifically, the location and appearance of an object in a sequence of images reflects its behavior, and this sequence of observations forms a predictable history.

Tracking the activity of objects can have an impact on the vision system by confirming or disconfirming the classification derived by the vision system. If the vision system misclassifies an object, then the situation assessment has a chance to catch the error by detecting that the object is not behaving in a normal manner. Also, the ultimate goal of the system is to keep track of the activities of objects.

The method of monitoring the situation is to use "scripts" [12]. Scripts provide the capability to predict future behavior and infer unobserved behavior. The scripts provide a knowledge structure to match against observations. The observations are matched to a consistent set of scripts for all objects. The script is flexible in the sense that not all events of the script have to be matched, so if the object is not observed at one stage the script will still match.

In the desired application, there are several ambiguities that make it necessary to carry along multiple possible interpretations of the script. The ambiguities are kept until only one interpretation is consistent. If no interpretations are consistent, then an anomalous situation has occurred, and it is reported.

A prototype script interpretation system has been designed and implemented on a Symbolics LM-2 LISP machine. Following evaluation and refinement, the script system will be translated into FRANZLISP and rehosted onto the VAX.

3.5 SYSTEM DEVELOPMENT TOOLS

As the need for modifications and extensions to ACRONYM became clear, so did the need for specialized software tools to aid in this process. ACRONYM already possessed several tools. These included a control trace, goal tree trace, and (remarkably) a special rule set which helps determine why subgoals not satisfied by the system failed. However, these tools were personalized to the needs and preferences of the original system developer, and differed from our broader needs based on a shallower understanding of ACRONYM implementation philosophy and history.

In keeping with our primary objective of demonstrating a prototype system used by programmers (as distinct from building a production system) it was decided to add only minimal facilities. A rudimentary explanation capability has been added in the form of the ability to interrupt (break) system operation upon encountering a designated rule or subgoal, and then ask "How" or "Why" the system was attempting what it was trying to do. System responses consist of subgoal and rule names and will soon be capable of including phrases describing the purpose of each (rule-controlled) step. Traces can be switched on or off by "executing" a designated rule or subgoal. The ability to display bindings to variables and data structures is also planned. These facilities will be further extended as specific needs arise.

4.0 CONCLUSION

At this point, the development and demonstration effort is going smoothly, and we are fairly optimistic about achieving our objectives. We expect to be able to demonstrate significant technical contributions in three areas: shadow prediction, model-directed segment extraction, and tracking of temporally related observations across a sequence of images. Good progress is being made in all three areas.

5.0 ACKNOWLEDGMENTS

Our task would have been much more difficult without the help and cooperation of Rod Brooks, the architect and developer of ACRONYM. He converted the MACLISP code to FRANZLISP and is providing consulting and programming support. Tom Binford and Barry Soroka have also made several helpful contributions. Andy Hanson was invaluable during our Eunice and ACRONYM rehost activities. Finally, Joe Mitchell provided programming support by writing the Nevatia-Babu linefinder code and later rehosting it to the VAX.

REFERENCES

[1] T.O. Binford, "Visual Perception by a Computer," IEEE Conf. on Systems and Controls, Miami (December 1971).

[2] R.A. Brooks, "Goal-Directed Edge Linking and Ribbon Finding," Proceedings: ARPA Image Understanding Workshop, (April 1979) pp. 72-76.

[3] R.A. Brooks, R.Greiner, and T.O. Binford, "The ACRONYM Model-Based Vision System," Proceedings IJCAI-6, Tokyo (Aug. 1979), pp. 105-113.

[4] R.A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," Artificial Intelligence 17, (1981), pp. 285-348.

[5] R.A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," Ph.D. Dissertation, Stanford University, 1981.

[6] R.A. Brooks, ACRONYM Manual, unpublished.

[7] M.A. Fischler, "The SRI Image Understanding Program", Proceedings: ARPA Image Understanding Workshop, (April 1980), pp. 71-88.

[8] M.A. Fischler and A.J. Hanson, "The SRI Image Understanding Program," Proceedings: ARPA Image Understanding Workshop, (April 1981), pp. 223-235.

[9] P.G. Mulgaonkar, L.G. Shapiro, and R.M. Haralick, "Recognizing Three-Dimensional Objects from Single Perspective Views Using Geometric and Relational Reasoning," Proceedings: PRIP, Las Vegas (1982), pp. 479-484.

[10] R. Nevatia and T.O. Binford, "Description and Recognition of Curved Objects," Artificial Intelligence 8, (1977), pp. 77-98.

[11] R. Nevatia and K.R. Babu, "Linear Feature Extraction and Description," Computer Graphics and Image Processing 13, (1980), pp. 257-269.

[12] R.C. Schank, Conceptual Information Processing. New York: North-Holland Publishing Company, 1975.

Appendix: DESCRIPTION OF ACRONYM

Acronym can be viewed as two major components: a basic core and the vision system. The core consists of support sub-systems based on artificial intelligence techniques, including a record package, a rule system, and a constraint system. This "tool kit" is very general in nature, and quite independent from any particular problem domain. Built on top of this core is the actual vision system. This consists of application-independent modules for modeling three-dimensional objects and matching these models to shapes extracted from two-dimensional images. Specific applications, depending on their difficulty, require various levels of application-dependent knowledge.

a.1 ACRONYM CORE

A major element of the core of ACRONYM is the record package. It provides a complete set of routines to handle the creation, maintenance, and access to a wide variety of data structures. The user can specify the form of each record structure, representing data either as simple list elements, or as named properties on an atom's property list. An interface to ACRONYM's LISP top level allows nodes of these data structures to be displayed. They are formatted as named slots with the data shown as fillers for these slots. Slot fillers can also be other nodes of the structure, thus permitting a complex graph structure to be formed.

The rule system, another major core element, provides the user with a means to write production system rules. These rules are written to conform to ACRONYM's rule format. This rule format specifies an advertisement, a set of pre-conditions, and an executable body. Once a rule has been written, it is translated by the rule parser into LISP source code, which is then compiled and loaded into the system. Also included are routines to initiate rules, trace their execution, and help the user debug rule sets.

A very powerful and useful element of ACRONYM is in the constraint system, which is capable of complex algebraic manipulations. The user may, for example, define some variable whose value is known only to lie within certain bounds. One element of the constraint system maintains these types of constraints in a normal form, using the record package described above. As the user (or some LISP function) defines these types of constrained variables, and begins to use them in algebraic equations, the constraint system will invoke other modules which perform algebraic simplification and determine bounds on values of the expression. As more constraints are entered or determined, the system tries to merge these constraints to determine tighter constraints either on variables or expressions. This provides a powerful, symbolic approach to algebraic aspects of problem computations.

a.2 ACRONYM VISION SYSTEM

The vision system is built on top of the ACRONYM core "tool kit". It is a powerful image understanding tool capable of matching three-dimensional object models to two-dimensional shapes extracted from sensed image data. Figure 9 is a block diagram of this system, viewed as four distinct modules which operate in both a top-down and a bottom-up fashion. From a top-down perspective, objects of interest are represented by a modeling system. From these models, the prediction system symbolically predicts two-dimensional shapes which may be expected to occur in imagery. From the bottom end, the description module extracts two-dimensional shapes by edge detection and shape determination. These predicted and observed shapes are then matched by the interpretation module.

Three-dimensional objects are represented in ACRONYM by models built from three-dimensional generalized cones. Each cone is described by an axial spine along which some surface is swept. The surface may be varied as it is swept along the spine. Some examples of generalized cones are shown in Fig. 10.

The modeling system makes full use of the constraint system, allowing the creation of classes of models, with generic classes having very loose constraints, and more detailed dimensions describing specific sub-classes. An example might be a generic class of land-based vehicles, with sub-classes of wheeled vehicles and tracked vehicles, and more detailed sub-classes of trucks, jeeps, etc.

ACRONYM's prediction module is used to transform the three-dimensional cones of the object models into two-dimensional shape surfaces, such as would be seen in actual imagery. In order to do the necessary two-dimensional shape predictions, the predictor requires some knowledge of the viewpoint. In ACRONYM this is represented as a camera model, which contains information about the position, orientation, and focal ratio of the sensing camera.

The prediction process is rule-based, with specific rule sets having knowledge of the geometric reasoning processes required for shape prediction. Certain rules predict the classes of shapes that may be seen from the camera viewpoint, while others predict the spatial relationships between these shapes.

The prediction process also makes use of the constraint system, allowing predictions based on incomplete information about the camera model. Predictions are also constrained by uncertainties in the model specifications. The result of predictions based on loosely constrained camera parameters or generic model classes would be general descriptions of the possible shapes, with constrained ranges of dimensional values and spatial relationships.

The description module in ACRONYM attempts to extract edge-based shape features from imagery. Edge detection is performed by the Nevatia-Babu linefinder developed at USC. From this edge segment information, the ribbon finding process attempts to form closed, convex shapes. These are then described as "standard" shape description corresponding to the types of shapes generated by the prediction process.

Finally, the interpretation module matches shapes found by the description process to the predicted shapes, and reports these matches.

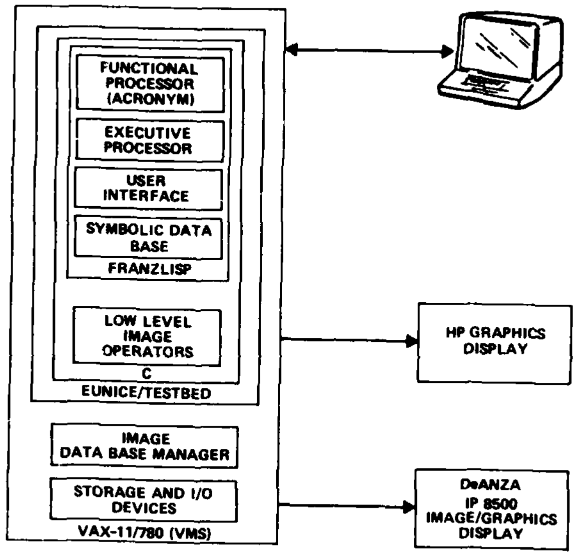


Figure 1. IU System Configuration.

11910-29

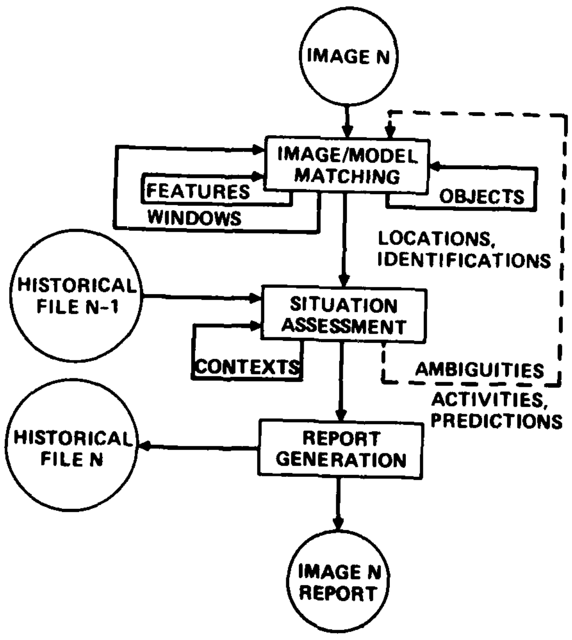


Figure 2. Overall IU System Functional Structure.

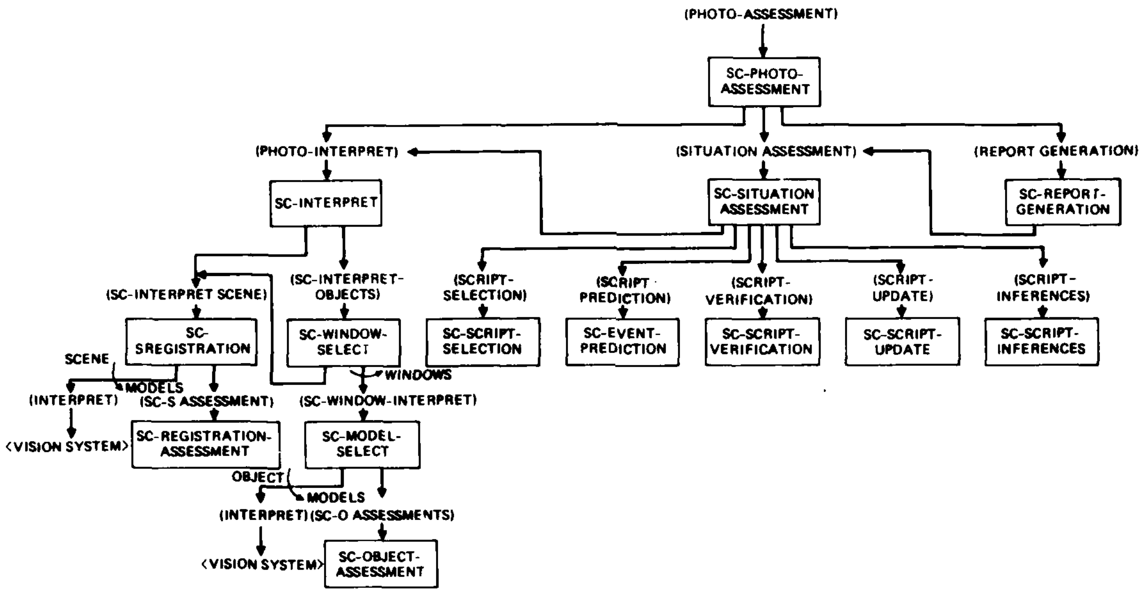


Figure 3. Top-Level Control Rule Hierarchy.

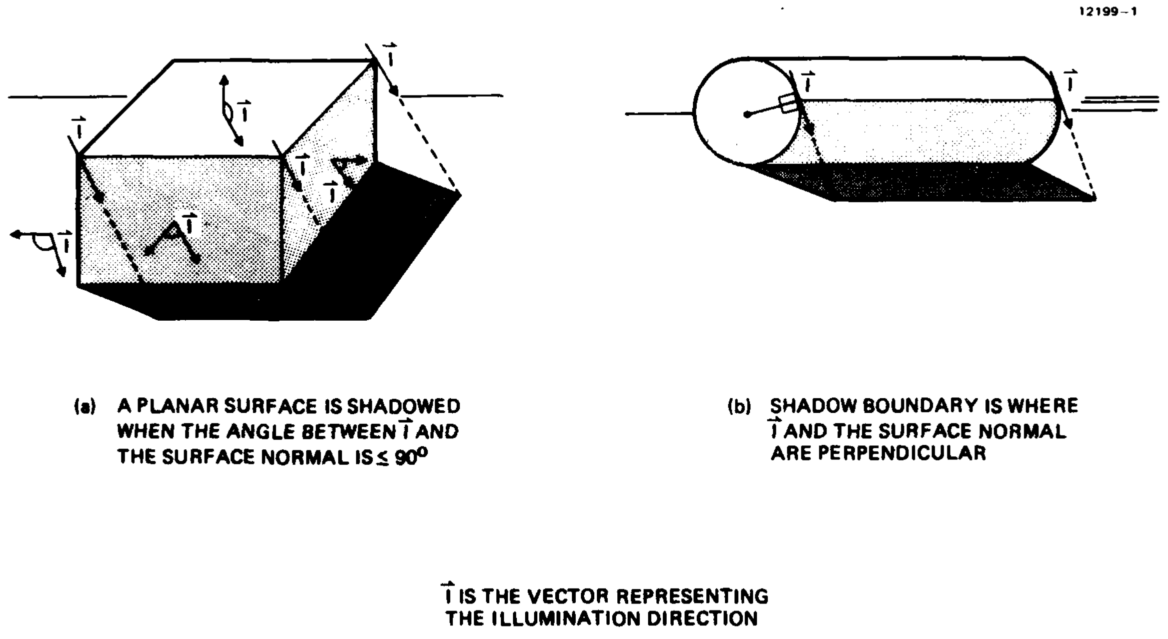
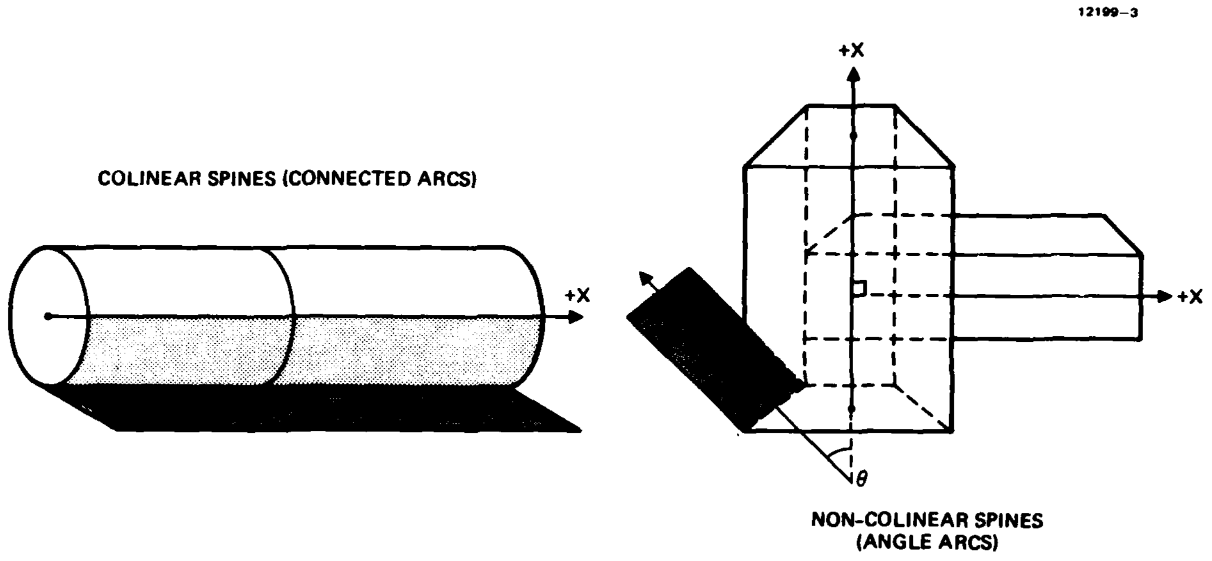
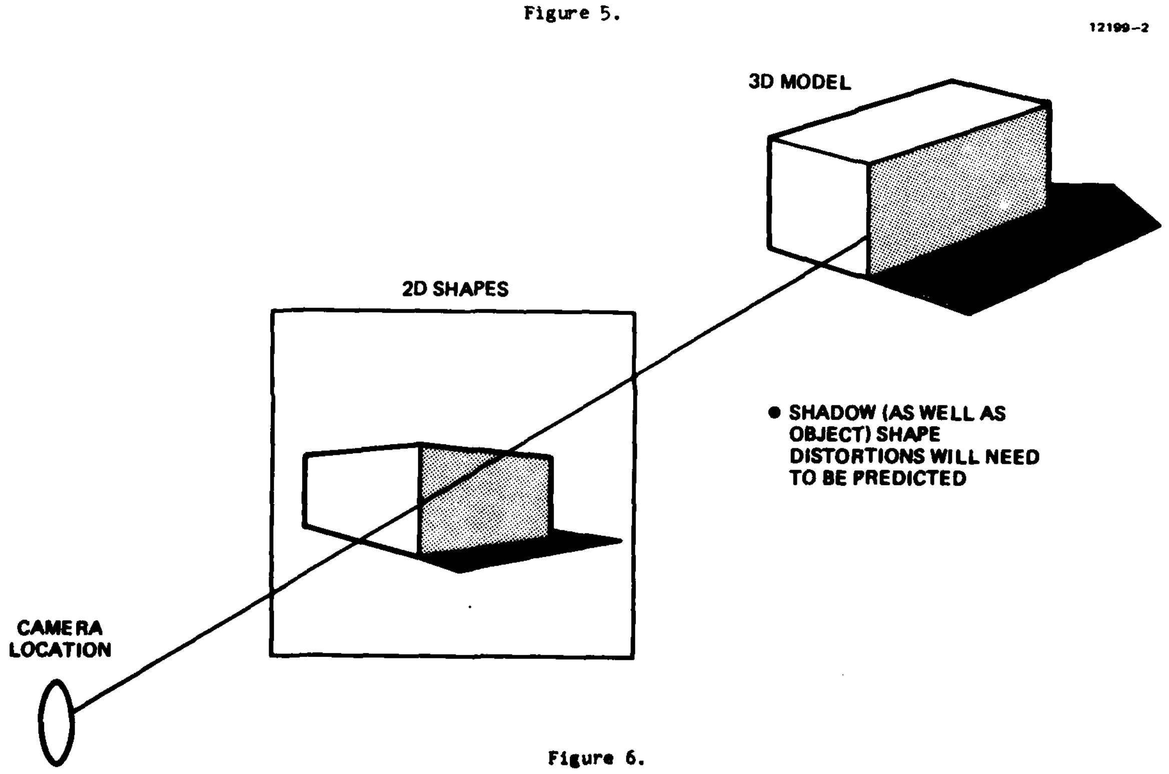

12199-1
(a) A PLANAR SURFACE IS SHADOWED WHEN THE ANGLE BETWEEN $\vec{I}$ AND THE SURFACE NORMAL IS ≤ 90°
(b) SHADOW BOUNDARY IS WHERE $\vec{I}$ AND THE SURFACE NORMAL ARE PERPENDICULAR
$\vec{I}$ IS THE VECTOR REPRESENTING THE ILLUMINATION DIRECTION

Figure 4.


12199-3
COLINEAR SPINES (CONNECTED ARCS)
+X
+X
+X
θ
NON-COLINEAR SPINES (ANGLE ARCS)

- SPATIAL RELATIONS BETWEEN SPINES OF SUBPARTS ARE PREDICTED
- SPATIAL RELATIONS BETWEEN THE SPINES OF OBJECTS AND THEIR SHADOWS WILL ALSO BE PREDICTED

Figure 5.



- SHADOW (AS WELL AS OBJECT) SHAPE DISTORTIONS WILL NEED TO BE PREDICTED

Figure 6.

12097-38

- HUERISTICS NEED KNOWLEDGE OF SHAPE, WIDTH, LENGTH, ETC TO "FILL IN" MISSING LINES
- SPATIAL INDEXING CAN EFFICIENTLY LOCATE PARTIAL SHAPES

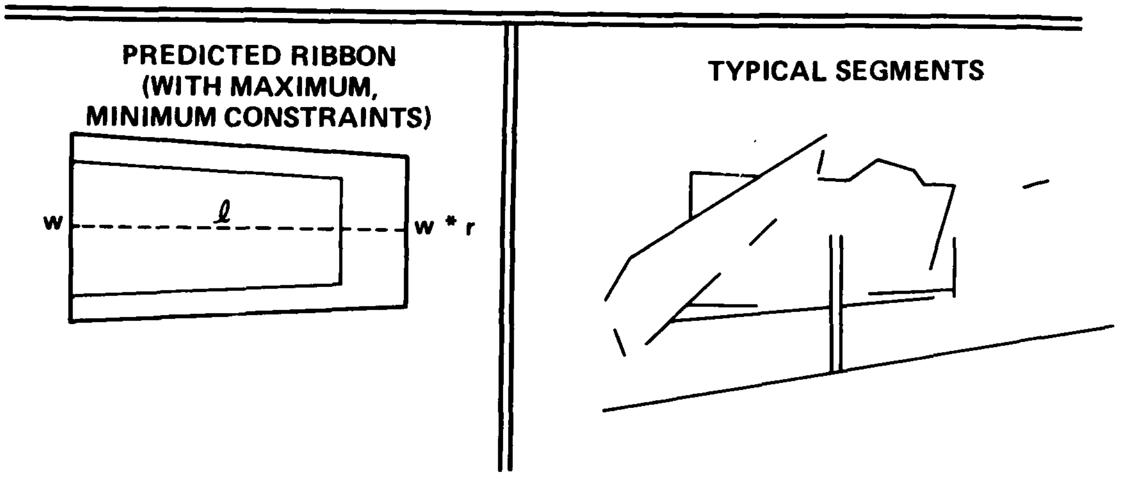


Figure 7. Model-Driven Shape Extraction.

12199-6

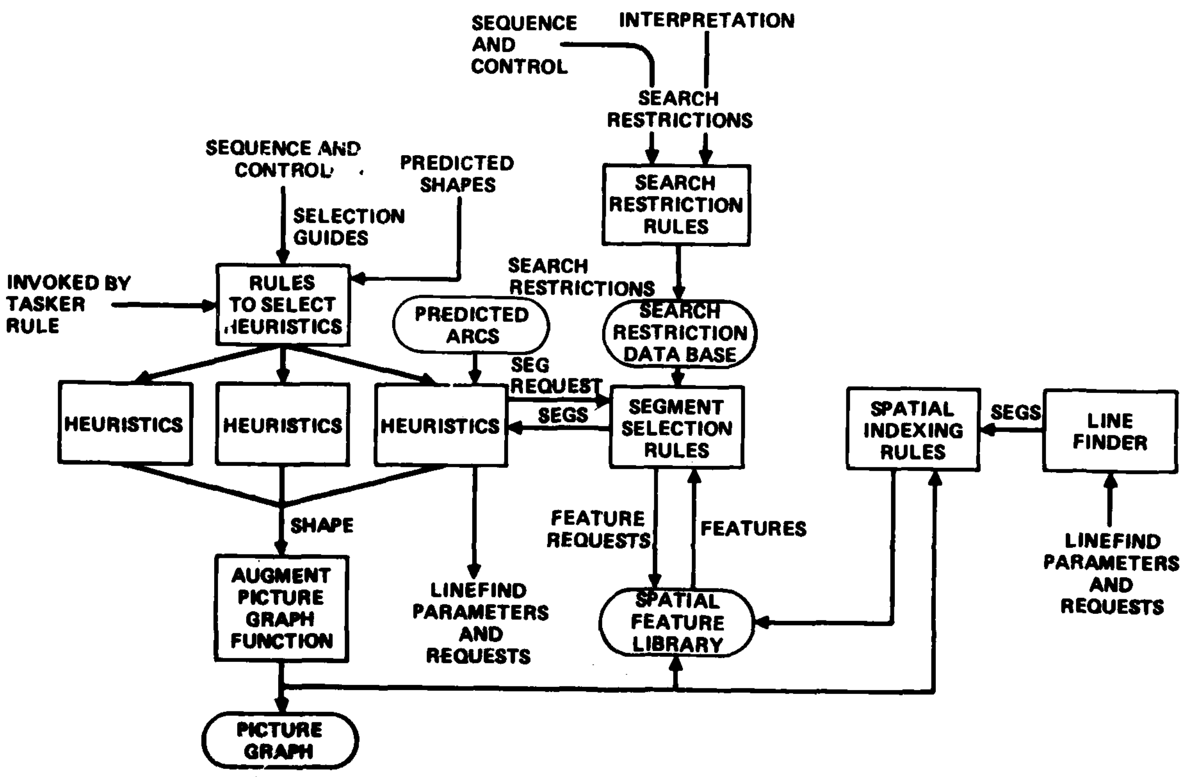


Figure 8. Shape Extraction Module Design Structure.

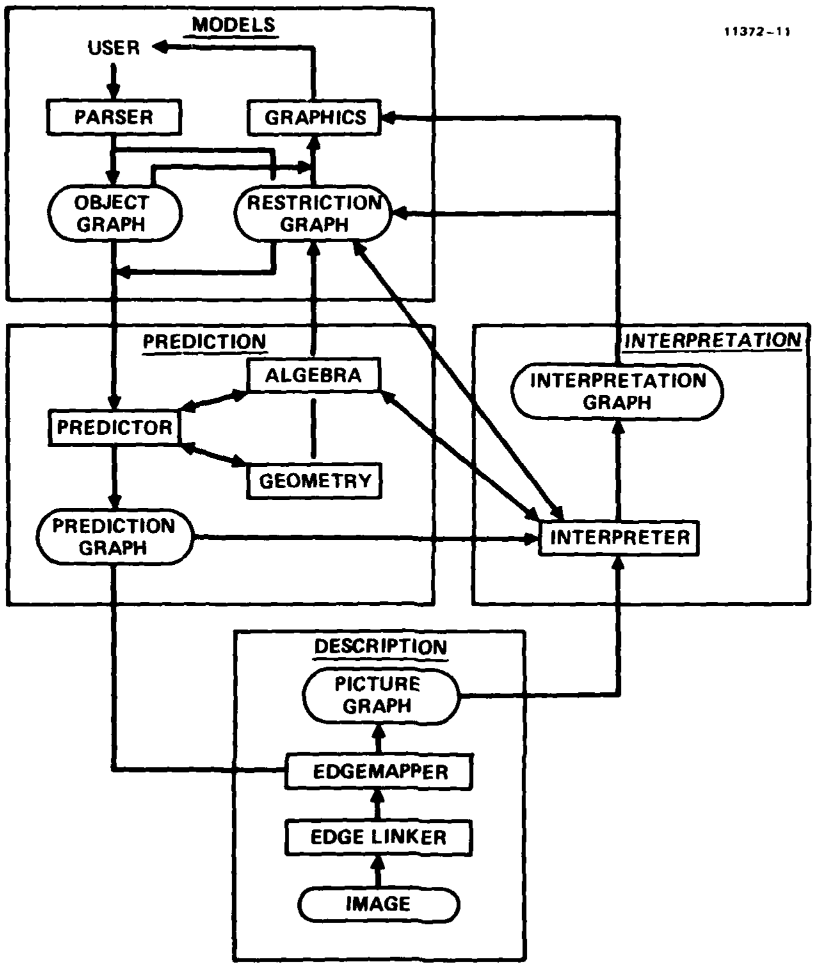


Figure 9. Acronym Vision System.

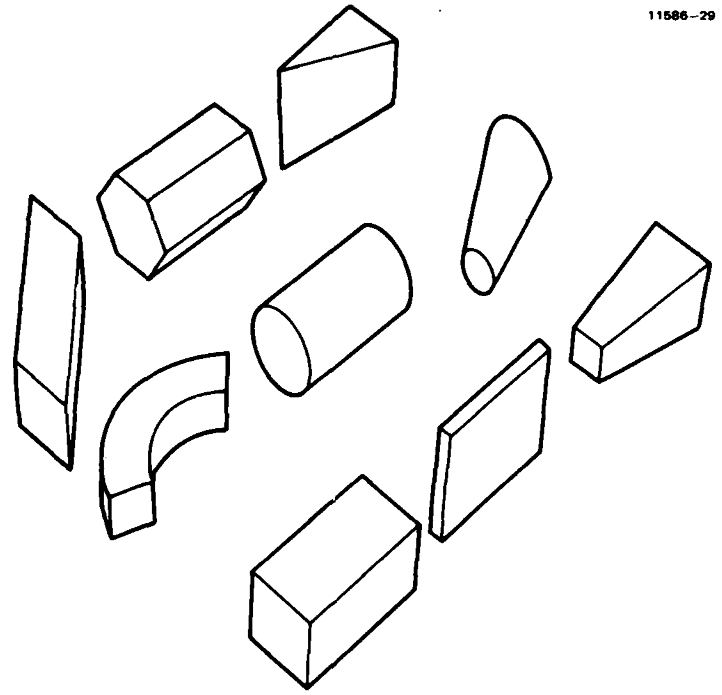


Figure 10. 3-D Generalized Cones.

AD-P000 110

RADIUS : A PROGRESS REPORT

S. D. Fouse and G. R. Nudd

Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, California 90265

ABSTRACT

A novel Residue Arithmetic Digital Image Understanding System (RADIUS) is described. This represents an application of VLSI technology to real-time image understanding. Extensive use is made of high speed programmable look-up tables to perform a wide variety of functions on a sliding 5x5 kernel. Residue arithmetic techniques minimize look-up hardware and provide effectively concurrent arithmetic. An interface to the PDP UNIBUS has been developed. Preliminary results of a study to determine functional requirements of a complementary logic processor are presented.

INTRODUCTION

The advent of Large-Scale Integrated circuit technology has radically changed the philosophy of computer design and engineering. Until recently, because of the extraordinarily high cost of computer development, machines were typically aimed at the widest possible user base. This has led to the almost exclusive development of the Von Neumann type of architecture, which is currently in use in general purpose machines today. Typically these computers have the advantage of general applicability, but for many applications they can be very inefficient. In certain areas the applications' importance and the processing needs are such that special purpose machines optimized for a single application would have significant advantages. Image Understanding and Artificial Intelligence are such areas where the processing and data access requirements are sufficiently stringent that novel architectures should be considered. The advent of Very Large Scale Integrated Circuitry (VLSI) and high-level design technologies have enabled such efforts to be seriously considered.

In addition to providing the potential for making more dense and higher speed structures, VLSI essentially provides an opportunity to re-think the conventional restrictions on computer architectures. An important part of this new philosophy is the realization that data access and interconnection are now the significant bottleneck in high speed processors, rather than the speed of the ALU. This has led to much work aimed at regularizing the flow of data through the processor and the elimination of long and circuitous data paths. Examples of this approach are contained in the Systolic and Wavefront approaches of H. T. Kung and S. Y. Kung. Further, the need for a regular and uniform topology on the semiconductor material itself has been recognized by experts such as C. A. Meade. The benefits of regular arrays of devices with, wherever possible, only local neighbor interconnect and communication, are both of higher device speed and hence increased throughput, and ease and feasibility of VLSI design.

Our work on the IU has recognized this, and we have wherever possible structured the architecture in terms of local, modular blocks of devices. Since the extreme of this approach is the memory array with large numbers of identical cells regularly interconnected, we have structured our processor around a look-up table. However, in order to provide the necessary dynamic range and flexibility for all low-level image understanding operations, we have decided to implement the arithmetic in 'residue' notation. Using this approach we have been able to provide a fully programmable processor which operates over a 5 x 5 kernel and performs all the commonly used arithmetic functions in image understanding using a memory block of only 32 x 5 bits. Significant additional advantages of this approach include extendability, selectable dynamic range, fault tolerance, and the ability for self-checking. We have developed a single 10,000 transistor LSI chip for this processor, as shown in Figure 1. We currently use 20 of these in the full processor. (An eventual VLSI device with 1 micron design rules and 80,000 transistors would perform the full operation.)

To complement this processor some form of logic processor is required, which typically requires less throughput and considerably reduced dynamic range. With these two machines (the RADIUS and logic) a full system consisting of DEC host, the RADIUS, and LOGIC can be developed. These issues are discussed below.

RADIUS

PROCESSOR DESCRIPTION

The Residue Number System

The residue number system was extensively studied by Tsabo and Tanaka[5]. It is based on a collection of N integers: $m_1, m_2, \ldots, m_N$; each of which is called a modulus.[1] The moduli must be relatively prime, i.e. have no common factors. The residue of x mod m_i is the least positive integer remainder of the division of x by m_i where x is the number to be converted and m_i is the modulus. Since the dynamic range of the full processor, M, is given by the product of the moduli, any integer in the range $0 \leq x \leq (M-1)$ can be uniquely represented. These residues are somewhat analogous to the digits in conventional numbers.

The strength of the residue number system lies in the way in which arithmetic operations are performed. Essentially arithmetic operations are performed in parallel in each base and then combined to uniquely determine the final result. An example of residue addition is given in Figure 2. Here we choose the moduli of 5, 7 and 8 giving a dynamic range M of 280. Thus the range before overflow is from 0 to 279. In the example, 19 is converted to 4, 5, 3 in the residue system. The 4 is the residue of 19 mod 5, etc., as shown. Each column is then independently added and the sum is expressed as a residue of the associated modulus. In the first column 4 + 2 = 6, which is equivalent to 1 modulus 5. The other columns are processed in a similar manner. The result 1, 1, 2 means that the actual result, when divided by 5, gives a remainder of 1, when divided by 7 a remainder of 1, and when divided by 8 a remainder of 2. These numbers can then be uniquely decoded to determine the result, 106, within the range 0 to 279.

As the example shows, the arithmetic operations performed on each modulus are independent of each other, making them ideally suited to parallel computation. Furthermore, look-up tables can be used to perform the operation on each modulus, giving very high speed operation. By using a sufficient number of moduli it is easy to produce an adequate dynamic range, yet the look-up tables can be kept small. There is a certain amount of overhead involved in converting from binary to residue and residue to binary, but the high speed of the arithmetic operations more than compensates for this.

Implementation of Algorithms

Conversion from a binary representation to a residue representation is straightforward. It requires that we calculate

$$(x \bmod m_1,\ x \bmod m_2,\ \ldots,\ x \bmod m_N)$$

where x is the value of the input data and m_i is the ith base. The simplest way to perform this calculation for a general set of bases is to use Read Only Memories (ROMs). By connecting the input data to the address lines of the ROM and looking at the ROM's data lines for the output, a look-up function is performed. For our particular processor which will support an eight-bit input dynamic range and bases which can be encoded in five bits or less, the size of an encoding ROM is 256 x 5 bits.

Once the data have been converted to residue format, we can perform a variety of calculations by using random access memory. For example, to do a convolution on the image data it is necessary to multiply each of the 25 elements of the kernel by the assigned value, and to add the results. The multiplications are done by using a programmable look-up table on each base of each number, 100 operations in all. Since the largest base is 32, no look-up table need be larger than 32 x 5 bits, and all look-ups are done in parallel.

Conversion back to binary format is based on the mixed radix method. In our system, we exploit the fact that we require an output dynamic range of only eight bits. The method can be explained by considering an iterative process where at every iteration the smallest base is eliminated by dividing the data value by that base value. Dividing essentially reduces the dynamic range of the value and thus eliminates the need for the extra base. Of course, since we are limited to a strictly integer system, we must make sure that the value is evenly divisible by the smallest base. This can be done by rounding up or down so that the element in the residue vector for that base is zero. An architecture that performs this mixed radix conversion for a four base system looks like the 3-level tree shown in Figure 3. At the bottom level of this tree structure the fourth base is eliminated. At the next highest level of the tree the third base is eliminated. Finally we are left with two base values which can be decoded with a simple look-up table.

Hardware Configuration (Data flow)

The actual computations on the image data are performed by a custom LSI NMOS circuit. The circuit processes a 5 x 1 kernel and is capable of performing computations of the form

$$y = \Sigma f_i(x_i)$$

where y is the output value, x_i are the five elements in the kernel, and f_i are polynomial functions of a single variable.

A schematic of the circuit is shown in Figure 4. The word size for this is five bits, which limits the bases used to a value of 32 or less. The circuit is designed to accept a 5-bit input word (at a 10 MHz rate) which is clocked into a 5-element shift register. The contents of each register element is then shifted to the next register. The data in each of the shift register elements is used to address a look-up table, which is a 32 x 5 Random Access Memory (RAM). This look up table performs operations requiring only a single operand such as multiplication by a constant or a squaring operation. The outputs of the 5 RAMs are then summed modularly to produce a 5-bit output, the base of the modular addition being programmable by external control of the circuit. Since the look-up tables are composed of RAMs the circuit can be programmed for many different computations, such as different weights for a convolution or different powers of a number for a statistical calculation.

To utilize this circuit (with a 5 x 1 kernel) in a 5 x 5 local area processor, multiple copies of the circuit are used as well as additional logic to combine the outputs of the individual circuits. For each base, 5 of these circuits are used, one for each line of the kernel. In addition, four 1024 x 5-bit ROMs are used to sum the outputs of the five circuits. Figure 5 shows a block diagram of the processor and Figure 6 is a picture of the actual processor.

UNIBUS INTERFACE

A most important feature of the RADIUS processor, in addition to its potential for large scale integration for military systems, etc., is that it can be attached to a general purpose machine in a research environment for the development of high level image understanding techniques. In fact, the high speed design (data are accepted at 100 nanosecond intervals) will also allow real time stand alone operation, but other considerations become important when the RADIUS processor is used as a peripheral device with a general purpose computer.

In order to get optimal use of the processor we need the fastest type of transfer available between the processor and the main memory, where the data to be processed will reside. The Direct Memory Access (DMA) type of transfer is the fastest type of transfer that a general purpose computer can support, since it does not require processor intervention. For DEC UNIBUS applications, the data rate is approximately 1 Megabyte/sec.

We have designed an interface around a DEC DR11B UNIBUS parallel interface card. This provides both the DMA transfer capability and several control lines to allow multiple transfer modes. To operate the interface the starting address and the amount of data to be transferred are specified, and a start code given. The DR11B then transfers the specified data without further intervention. The control lines specify whether look-up table data or image data is involved in the transfer. For look-up table data, the interface simply passes the data to the 16 bus lines in the processor. For image data transfers, the interface is more complex. Since the DR11B transfers 16 bits at a time, each word contains two eight-bit pixels. Following the word transfer, the interface splits it into two bytes which are then fed consecutively to the RADIUS processor. After transferring a complete line of pixels, the DR11B is switched into read mode and one line of processed image data (stored in the digital line delay) is read back into the computer.

Several diagnostic capabilities are built into the interface. The standard DR11B card can be tested alone, and the custom built part of the interface can also be tested once operation of the DR11B has been verified. Also, the contents of the look-up tables in the RADIUS processor can be read back into the computer to verify their accuracy.

The entire interface is built on wire wrapped cards which plug directly into the backplane of an 11/34. This connects to the RADIUS chassis with ribbon cable so that the entire assembly can be simply plugged into a UNIBUS. Furthermore the software is being configured so that the RADIUS processor can be called as subroutine. Consequently, any DEC computer user is able to do rapid 5 x 5 image processing operations with great ease. Since the UNIBUS operates at 1 M Byte/sec, a 256 x 256 image can be processed in 132 msec.

RESULTS

The RADIUS processor performs a variety of local neighborhood arithmetic operations using a 5 x 5 kernel. A typical function that can be programmed is a convolution or a sum of products over the 5 x 5 window. To program the processor to perform the convolution, the 25 weighting factors are selected, and a multiplication look-up table is generated for each weighting factor for each of the four bases. The only two constraints in the selection of the weighting factors are that they be integers and that the combination of the dynamic range of the weights and input data not exceed the internal dynamic range of the processor, approximately 2^{18}. These look-up tables are transferred into the processor, which is then capable of performing the programmed convolution at input pixel rates up to 10 MHz.

A convolution is just one of the possible operations that can be performed. Another useful function for image processing that RADIUS is capable of performing is the sum of squares operation. This is used frequently in the calculation of variances and other statistical moments which are commonly used for texture discrimination. To program the processor for sum of squares one calculates a squaring table for each base and transfers these tables into the processor. In general, the processor can evaluate an integer coefficient polynomial function for each element in the 5 x 5 local area and then sum the results for each of the 25 functions. Table 1 lists some of the functions that RADIUS can perform.

A typical example of the processing capability of the architecture is shown in Figure 7. Here we show an input scene (a) which has been decomposed into six (b through g) at angles $n\pi/6$ (for $n = 0$ to 5). Each of these resulting images provides a calculation of the edge magnitude for a particular direction, and in this sense a full edge evaluation has therefore been completed. This is most important for any subsequent line finding for object identification or image registration, etc. A commonly used and effective technique based on the Nevatia Babu concept is to perform these 6 separate edge calculations for each pixel in the original image as shown, and then to select, again for each pixel, the largest edge magnitude. This therefore provides the dominant edge vector for each location in the image, and can be readily used in the subsequent thinning and connectivity algorithms for full line tracing. This vector edge composite as calculated by the processor is shown in Fig. 7(h). The form of this particular example is a linear sum of products or convolution, but it should be emphasized that other non-linear operations such as variance, standard deviations, etc., can similarly be performed using RADIUS.

LOGIC PROCESSOR

As has been said many times before, the main computational bottlenecks in image analysis are operations which are performed in the pixel domain. There have been many processors which have been developed which can do very fast operations in the pixel domain, but they are typically constrained in the type of operations they can perform. Since the development of pixel level operations is still a very active field it seems only reasonable that new processors will be required to implement these new operations.

PROCESSOR REQUIREMENTS

Many operations that are performed on every pixel have a fixed data flow and require no logic or decision capability to be performed. For these type of operations a processor such as RADIUS is ideally suited. However there are many operations which are performed at the pixel level which do require some sort of logic or decision capability. There are processors which are suited to these logic operations but they have not been designed to handle the sophisticated operations that are being developed and refined in the image understanding community today. These operations include edge thinning, edge linking, shrink and swell algorithms, border following, and many more. We have examined some of these operations to determine what primitive operations will need to be supported in a new generation logic image processor.

Edge Thinning

The edge thinning algorithm we chose as a representative example is the Nevatia-Babu thinning that is incorporated in their line finder system. The basic algorithm is as follows :

1. Form a 3x3 neighborhood around the edge point of interest.
2. Get the direction of that edge.
3. Extract the two neighboring edge points which are perpendicular to the direction of the center edge.
4. Perform tests (comparisons) on the directions of the two neighbors and compare their magnitudes to that of the center pixel.
5. Replace the center pixel with 0 if the tests are false, otherwise leave alone.

We find several types of operations in this thinning algorithm. First of course is the neighborhood formation, which is certainly expected but bears mentioning. Second is the use of an additional input, edge direction, to control or select the operation. Third is the tests which are performed on both the edge direction and edge magnitude for both the center pixel and the two selected neighboring pixels. Finally the actual function determination and pixel replacement.

Edge Linking

The algorithm we have chosen to be representative of this type of operation is the linking portion of the Nevatia-Babu line finder, which is performed after the edge thinning. This linking operatior has been designed with two phases. The first phase calculates a predecessor and sucessor for each edge point. The second operation then constructs the lists of connected edges. The first phase of the linking is well suited to a raster oriented pixel processor, while the second phase requires a random access capability to the data, so we only considered the operations in the first phase.

This linking operation is more complicated than the thinning operation in that comparisons must be made on both the edge directions and magnitudes. Successor candidates are generated by comparing the direction of the eight neighboring pixels to the direction of the center pixel. A neighbor is a possible candidate if its direction is within thirty degrees of the center pixels direction. To select among several possible successors their individual directions must be compared as well as their magnitudes.

Coarse to Fine Operation

The combination of shrink and swell functions can be used to look at the image at different resolutions, i.e. go from fine to coarse. This is useful for deleting small components from images, calculating "skeletons" of components, determining sparsity of components, etc. The simple operations of shrink and swell are as follows :

Swell :	Change 0's to 1's if they have any 1's as neighbors.
Shrink:	Change 1's to 0's if they have any 0's as neighbors.

The major functions performed are the generation of the neighborhood, a test for 1's or 0's in the neighborhood, and a replacement operation which depends on the outcome of the test. This operation can be enhanced by using a slightly more complicated test on the neighborhood. By determining if a point is simple, i.e. its neighborhooa has exactly one point adjacent to p, then the shrink and swell operations preserve component connectedness in the image, and can be used to count connected components in the image.

Border Following

Rosenfeld and Kak describe a border following algorithm[6] which is an iterative labeling scheme, but which uses random access of the pixels. Presumably similar methods can be constructed that would have comparable performance but could use raster access to the pixels. The algorithm works by creating a seed to the border and then propagating the border by looking at a subset of the neighbors and either stopping on some criterion or setting the value of the center pixel to a border value and selecting the next center pixel. In a raster algorithm additional logic would be required to determine if the pixel was a border, but this logic should be comparable to that of selecting the next border point. The types of tests performed by the algorithm include comparing the center pixel to a constant (3), comparing neighbors to two constants 2 and 4 and selecting the first neighbor (in a clockwise rotation) that is nonzero.

Zero Crossings

This operation is well known for its use in the generation of the primal sketch, whereby a functions first derivative is maximized by finding zero crossings in the second derivative. To perform this operation one neeYs to look in the eight directions from the center pixel and determine if there is a change of sign. This implies that the processor is able to form the 3x3 neighborhood, represent positive and negative values, and compare signs.

We have examined five different functions which require different kinds of logical operations to be performed on image pixels. Table 2 summarizes the set of primitive operations that would be required of a processor which could perform all of the functions we have examined.

SUMMARY

We have described a novel processor (RADIUS) which uses residue arithmetic to achieve the high throughputs required for real-time image understanding. This processor is currently implemented using a custom LSI circuit but is ideally suited to a VLSI implementation because of its memeroy intensive architecture. In addition we have developed an interface which allows RADIUS to be attached to a general purpose machine and be used to speed up processing times of complex IU software systems. A natural extension to RADIUS, which does the computationally intensive operations, is a processor capable of performing logic operations on a grey level image. We presented preliminary results of a study to determine the functional requirements of such a processor. This study will be extended to include a wider variety of systems as well as better determining the needs of the IU community as a whole.

REFERENCES

[1] G. R. Nudd, "Image Understanding Architectures." National Computer Conference, May 1980, Anaheim, CA. AFIPS Conf. Proc. Vol. 49, pp. 377-390.

[2] S. D. Fouse, G. R. Nudd, and P. A. Nygaard, "Implementation of Image Pre-processing Functions Using CCD LSI Circuits." Proc. Society Photo-Optical and Instrumentation Engr. Vol. 225 pp. 118-130, Spie Conf. April 1980 Washington, D. C.

[3] R. Nevatia and K. R. Babu, "An Edge Detection, Linking, and Line Finding Program," USCIPI Report No. 840, Sept. 1978.

[4] K. I. Laws, Textured Image Segmentation, Ph.D. Thesis, USC, Electrical Engineerng Dept., January, 1980.

[5] N. Szabo and R. Tanaka, Residue Arithmetic and its Applications to Computer Technology, McGraw-Hill, New York, NY, 1967.

[6] A. Rosenfeld, A. Kak, Digital Picture Processing, Academic Press, New York, New York, 1976.

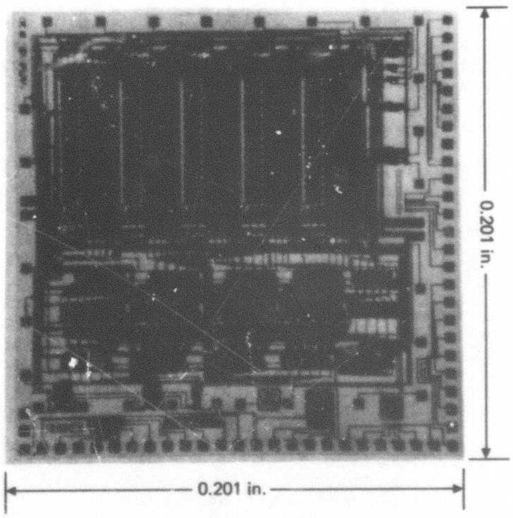


Fig. 1. Photograph of Residue LSI circuit

MODULI	5	7	8
19 ⟶	4	5	3
+87 ⟶	+2	+3	+7
106 ⟵	1	1	2

Fig. 2. Example of Addition Using Residue Arithmetic

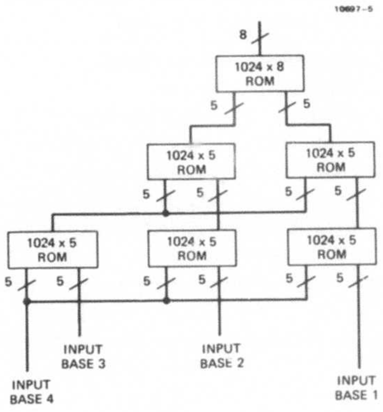


Fig. 3. Mixed Radix Residue Decoder

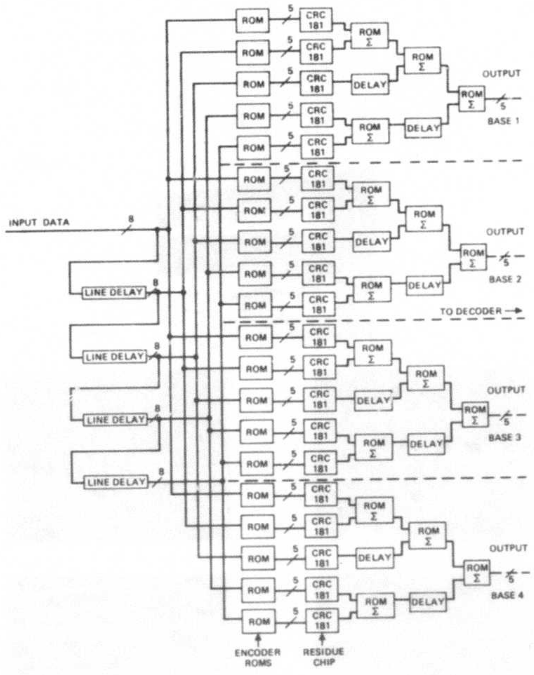


Fig. 5. Block Diagram of RADIUS Processor

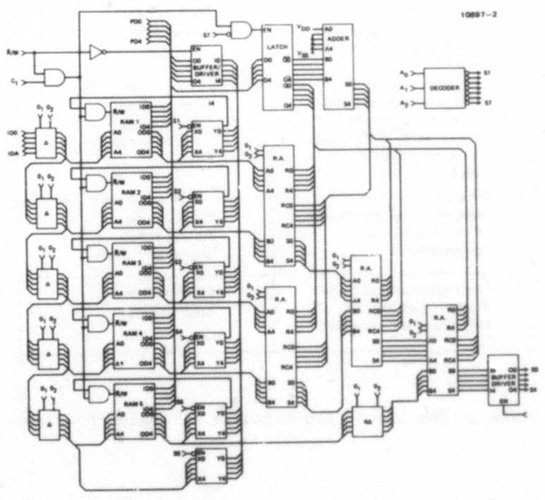


Fig. 4. Schematic of Residue LSI Circuit

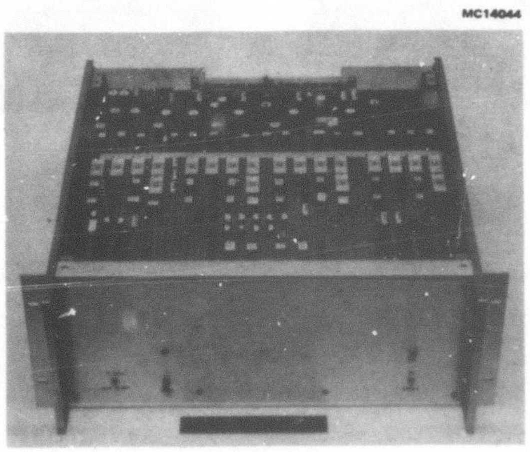


Fig. 6. Photograph of RADIUS Processor

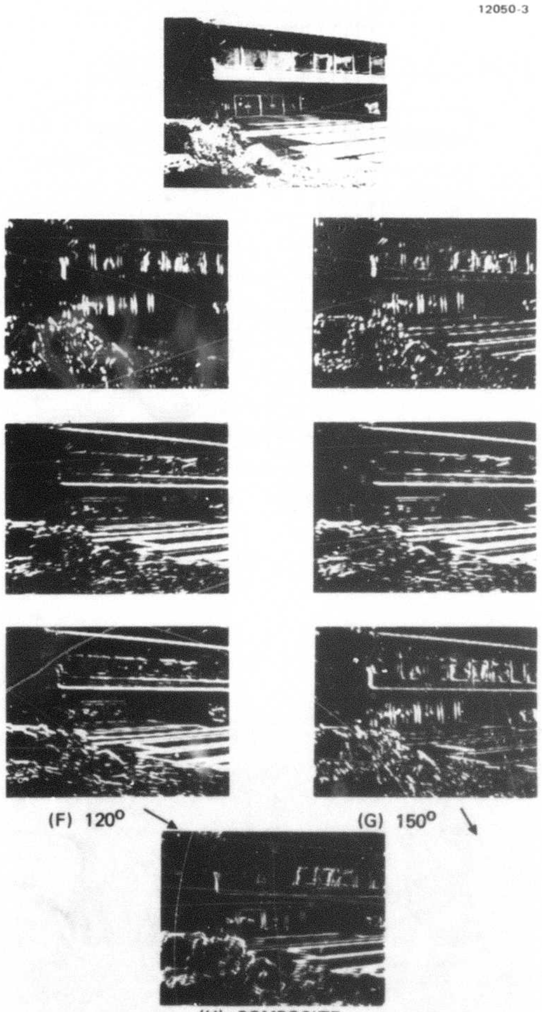


Fig. 7. Example of RADIUS applied to Edge Detection

POINT OPERATIONS
POLYNOMIAL FUNCTIONS
CONTRAST ENHANCEMENT
1– DIMENSIONAL OPERATIONS
INTEGER COEFFICIENT TRANSFORMS
POLYNOMIAL FUNCTIONS
2– DIMENSIONAL OPERATIONS
EDGE ENHANCEMENT
STATISTICAL DIFFERENCING
LOW PASS/HIGH PASS FILTERING
SHAPE MOMENT CALCULATIONS
STATISTICAL MOMENT CALCULATIONS
INTEGER COEFFICIENT TRANSFORMS
TEXTURE ANALYSIS

Table 1. Functional Capabilities of RADIUS

12186–1

PRIMITIVE FUNCTION	ALGORITHMS WHICH USE IT
Form neighborhood	All
Compare neighborhood values (including center) to a constant or multiple constants	Edge thinning, border following
Compare neighborhood values to each other or to a range determined by other neighborhood values	Edge thinning, Edge linking, Zero crossings
Determination of adjaceny (4 or 8) or simple connectedness	Edge linking, Border following, Shrink and swell
Use neighborhood from multiple pictures, i.e., magnitude and direction	Edge thinning, Edge linking

Table 2. Functional Requirements of Logic Processor

AD-P000 ## 1

A SURVEY AND EVALUATION OF FLIR TARGET DETECTION/SEGMENTATION ALGORITHMS

B. J. Schachter

Westinghouse Electric Corporation
Systems Development Divisions
Box 746, Mail Stop 440
Baltimore, Maryland 21203

ABSTRACT

A study was conducted at Westinghouse and the University of Maryland to survey and evaluate FLIR target detection/segmentation algorithms. The study was conducted in software on a data base of 50 FLIR images supplied by the three branches of the armed services. The study concluded that three techniques (double-window filters, spoke filters, and border followers) perform fairly well and are implementable in real-time hardware.

INTRODUCTION

This report describes results of a study conducted both at Westinghouse and the University of Maryland under the DARPA Image Understanding Program. One objective of the study is to survey and evaluate FLIR target detection/segmentation algorithms. Candidate algorithms were chosen from those developed at Westinghouse, the University of Maryland and other academic, governmental, and industrial organizations. Algorithms from the initial list were evaluated over a common data base if and only if: they passed certain preliminary tests, performed well in previous studies, or were claimed to perform exceptionally well by their authors.

Each algorithm was tested by its author's organization on the organization's own computer facility. The study had one rule: "No algorithmic parameters could be changed by human intervention during the evidential run through the data base.*" Refinements could be made to the software before this final pass through the data base.

DATA BASE COMPILATION

A data base of 50, 128 × 128 pixel FLIR images was compiled. Several images from the data base are shown in Figure 1. The sources of the images were four larger data bases prepared by the three branches of the armed services. Twenty of the 50 images were constructed from 64 × 64 pixel target images which were repeated four times by reflecting them about horizontal and vertical axes (e.g., see Figure 1b). These were used to test the algorithms' sensitivity to the orientation of targets.

The data base was compiled with a goal of diversity. Often results are reported in the literature in which an algorithm is tested on a very small collection of similar images. This allows an algorithm to be finely tuned to the ensemble statistics. In compiling a diverse data base, it was hoped to simulate realistic conditions in which little a priori information is available.

EVALUATION APPROACH

The data base was carefully hand segmented to obtain the centroid (C_I, C_J) and vertical (R_I) and horizontal (R_J) radii of each of the targets (Figure 2). This hand segmentation was a cooperative effort of several persons who had knowledge of the data base contents. All algorithms were required to use this same format for output. That is, for each detected target, in each image, the output is an estimate: ($\hat{C}_I$, $\hat{C}_J$, $\hat{R}_I$, $\hat{R}_J$).

A target is said to correctly *detected* iff:

$$\{|C_I - \hat{C}_I| \leq ½R_I + ½ \text{ and } |C_J - \hat{C}_J| \leq ½R_J + ½\}, \quad (1)$$

where all measures are in image picture elements (pixels). A *false alarm* is any detection outside this prescribed region. Extra detections inside the prescribed region are each scored as 1/3 false alarm.

With reference to Figure 3, let B denote the rectilinearly oriented box hat enclosing a target as determined by hand segmentation. Let $\hat{B}$ denote its estimate as output by a computer program. Segmentation accuracy is estimated by:

$$A = \frac{|B \cap \hat{B}|}{|B \cup \hat{B}|} \quad (2)$$

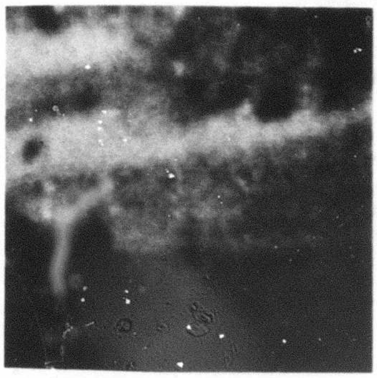
or the common area divided by the total area of the figures, averaged over detected targets. Note: $0 \leq A \leq 1$.

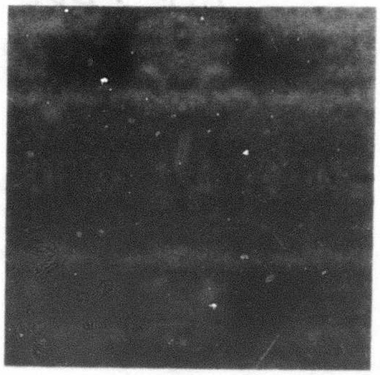
TESTS PERFORMED BY COMPUTER SIMULATION

One goal of this project is to determine the suitability of various hardware implementations of the candidate algorithms. This entails an evaluation of both performance and implementability. However, the tests discussed here took place only in software and did not take advantage of the additional information which would be available to a real-time target acquisition system, such as range to center of field of view. This point will be expanded upon below.

A target acquisition system typically consists of a number of pipelined stages. The first stage accepts images from the FLIR and does some preprocessing. This Preprocessor usually takes the form of one or more local filters which reduce noise, extract local features, and/or increase the contrast between targets and background.

* Note: Image polarity was allowed as an input.


(a) No Targets


(b) Jeep Repeated Four Times

(c) Tank

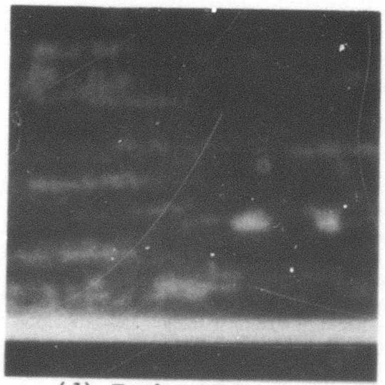

(d) Tank, APC

Figure 1. Four Images From Compiled FLIR Data Base

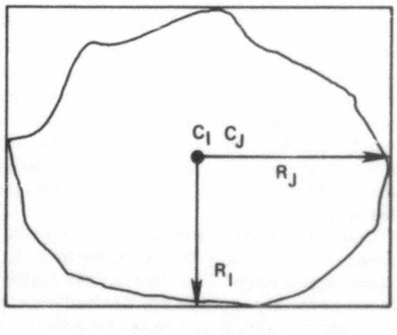


Figure 2. Illustration of Target Location and Extent Parameters

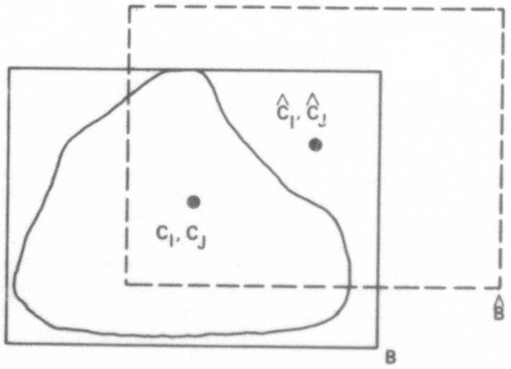


Figure 3. Illustration of Relation between Boxes Enclosing Actual and Detected Targets

The second stage is the Detector, which locates blobs it suspects of being targets. The centroids of these candidate targets are then passed to a Segmentor and/or Feature Extractor. This third stage either determines the borders of the detected blobs or extracts features to describe them. This information is passed to the fourth stage, called the Classifier. This final stage classifies potential targets, prioritizes them, determines confidence probabilities, and outputs this data along with target locations.

A real-time system mounted on a moving vehicle should take advantage of the following types of additional information which are usually available.

(1) *Range* - The range to the center of the field of view will usually be known. This will permit calculation of expected target size in pixels at that location. Size in pixels at other locations in the field of view can be estimated from assumptions or knowledge of terrain geometry.

(2) *History* - Past values of any computed parameters can be stored. (For example, the location of a detected target on past frames can be used to predict current location.)

(2) *Control Loops* - Feedback loops can be set up between any stages. (For example, the output of the Classifier can be used to control thresholds in the Detector.)

If a real-time system fits the model described, then separate circuit boards might be used for each stage. Boards implementing different algorithms could be interchanged, and the algorithms tested on large quantities of data. If board pin-outs and protocols are standardized, then even boards developed by different companies could be interchanged and tested in a competitive manner. No such standardization now exists. Furthermore, in current real systems, the separation in functions between the various stages is often not clearly defined. An algorithm implemented on a board may combine parts of several stages. Even the basic hardware organization may be different from that discussed above, with some stages operating in parallel or in a different order than indicated. There is always a tradeoff in performance between the various stages, since it is only the output of the final stage which really matters. For example, it is acceptable for a detector to operate at a high false alarm rate if a classifier will later cull out counterfeit targets. This is an important qualification for a study such as this, where individual algorithms are being tested, not entire systems. We will deal in part with this concern by plotting detection rate vs. false alarm rate, though our results are not as comprehensive as may be desired.

CLUSTERING IN MEASURE SPACE

The standard method of segmenting an image is by gray level thresholding. Here the classes correspond to gray level ranges, e.g., "light = hot" and "dark = cool". Since these ranges are not known in advance, they must be determined by examining the gray level histogram and looking for peaks (one dimensional clusters), and choosing thresholds (one dimensional decision surfaces) that separate the peaks.

A number of investigators have suggested that multidimensional feature space should also be useful for segmenting complex gray scale images. A variety of features may be defined over a neighborhood set about a pixel, e.g., mean, median, variance, commonality, total variation. This approach could be employed when a single feature, such as gray level, is not adequate for segmentation because the given image contains a number of textured regions whose gray level ranges overlap.

Initial work at Westinghouse has indicated that thresholding by cluster detection in histograms is not adequate for separating targets from background. Gray level target and background clusters are often not separable, i.e., their probability densities overlap. Likewise, the response of local operators tends to be rather variable, not yielding well defined clusters. The basic weakness of segmentation schemes which use only local feature values is that they attempt to classify image parts without regard to their relative positions in the image. It should not surprise us that any approach which does not take spatial contiguity fully into account fails much of the time.

ALGORITHMS SELECTED

The segmentation algorithms investigated in our study are those that make use not only of similarity but also proximity. The candidate algorithms have been classified into six groups:

(1) Double Window Filters
(2) Spoke Filters
(3) Border Followers
(4) Relaxation Algorithms
(5) Pyramid Approaches
(6) Mode Seekers

Each of these approaches will be described below. At least one method of each type will be tested on the assembled data base.

DOUBLE WINDOW FILTERS

A double window filter slides two non-overlapping windows over an image. The windows are both commonly rectangles, with one surrounding the other (Figure 4). The intensities within the inner window are viewed as samples of a random variable X and those of the outer window as samples of a random variable Y. The objective is to determine if the inner window surrounds a target, while the outer window contains background clutter (Figure 5). Since little a priori information is available about target and background statistics, some assumptions are usually made before the problem is formulated and solved. One way to pose the problem is in the language of statistical hypothesis. Doing so usually involves choosing distributions to model the behaviors of X and Y. Results, of course, are valid only if the chosen distributions correctly describe the experimental situation, namely that they provide the correct statistical model. Although *simple* statistical hypotheses concern only the parameters of assumed or known distributions, it is also possible to define *composite* hypotheses which also concern the fundamental forms of the distributions themselves. In either case, to construct a criterion for testing a given statistical hypothesis requires the formulation of an alternate hypothesis. Symbolically, we will let H_0 stand for the null hypothesis and H_A stand for the alternate hypothesis.

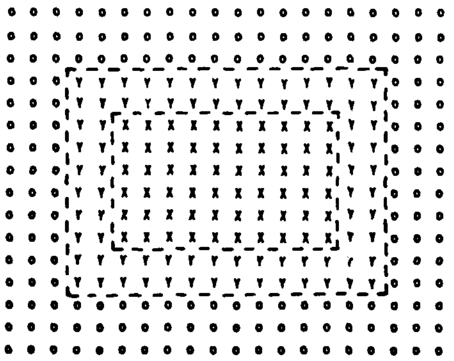


Figure 4. Window Geometry

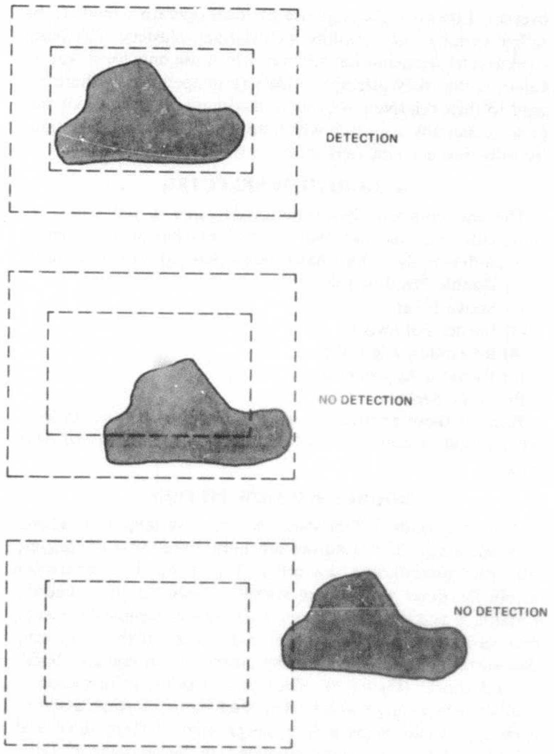


Figure 5. Three cases of target/filter relationship: (a) target completely within inner window, (b) target partially within inner window, (c) target outside inner window.

A typical simple hypothesis is as follows:

$X_1, \ldots, X_m$ is an independent random sample of a normal random variable X with mean μ_X and unknown variance σ^2_X

$Y_1, \ldots, Y_n$ is an independent random sample of a normal random variable Y with mean μ_Y and unknown variance σ^2_Y

$$H_o: \mu_X = \mu_Y$$
$$H_A: \mu_X \neq \mu_Y$$

This test can be performed in terms of a quantity T which has a t-distribution with m + n - 2 degrees of freedom [18]:

where

$$T = \frac{k(\overline{X} - \overline{Y})}{\sqrt{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2}}$$

$$\overline{X} = \frac{1}{m}\sum_{i=1}^{m} X_i, \quad \overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i. \tag{3}$$

H_o is rejected at a significance level α if and only if $t \geq c$, where t is the experimental value of T. If, for instance, $n = 10$, $m = 6$, and $\alpha = 0.05$, then $c = 2.145$.

If targets are always known to be hotter (i.e., of higher intensity level) than the background, the test can be reformulated as:

$$H_o: \mu_x = \mu_Y$$
$$H_A: \mu_x < \mu_Y \tag{4}$$

This approach and more complex formulations are conventionally used by Westinghouse in detection of targets in radar. A number of companies are using it for target detection in FLIR imagery.

Texas Instruments [TI] and Ford Aerospace [FA] have developed filters of this type. TI [2] uses a metric of the form:

$$C = \frac{(\overline{X} - \overline{Y})^2 + \hat{\sigma}_X^2}{\hat{\sigma}_Y} \tag{5}$$

FA [1] divides the outer window into N subregions. They use a metric of the form:

$$C = \sum_{k=1}^{N} (I_X - I_Y(k)), \tag{6}$$

where I_x is the mean of inner window pixels exceeding a specified threshold and I_Y (k) is the mean of outer window pixels in subregion k exceeding the threshold. Both TI and FA use range for the control of window size. Neither referenced paper provides a statistical model for the experimental design.

Woolfson of Westinghouse sees no justification in assuming normality or any other statistical distribution. He suggests an approach based on estimates of the probability density functions of random variables X and Y (as obtained from their sample values). A computer program was written to test this concept. The upper three bits of sample intensity levels were used to estimate probability density functions f_X and f_Y. Since three bits were used, these estimates in effect were quantized to eight entries each; i.e., $\hat{f}_X = (\hat{f}_X(1), \ldots, \hat{f}_X(8))$, $\hat{f}_Y = (\hat{f}_Y(1), \ldots, \hat{f}_Y(8))$. A target is detected if and only if the maximum intensity level in the inner window exceeds that of the outer window and

$$\sum_{i=1}^{8} |\hat{f}_X(i) - \hat{f}_Y(i)|$$

exceeds a given threshold. Since range was not available for these experiments, five filters were used in order of decreasing size, with a detection occurring upon first exceeding the threshold. Since several detections often occur over nearby pixels, detections were spatially clustered and replaced by their cluster centroid. The associated window widths and heights were averaged to form an estimate of target extent. Results are shown in Figures 6 and 7. The method works fairly well. It has difficulty with targets near image borders, but this is not a serious problem in an actual target acquisition system where the input imagery is typically 875 x 875 pixels in size and targets cover a relatively small area. Since this method uses raw gray levels, rather than edges, it is bound to produce a fairly stable output from frame to frame.

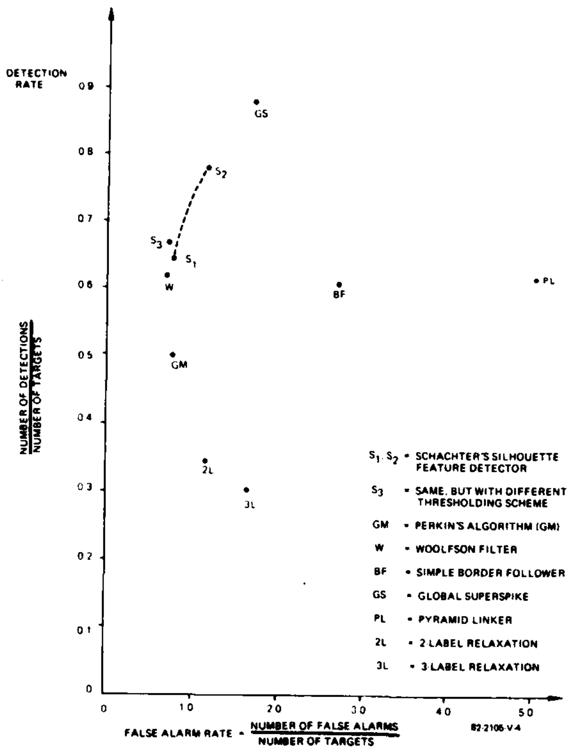


Figure 6. Detection Rate vs False Alarm Rate

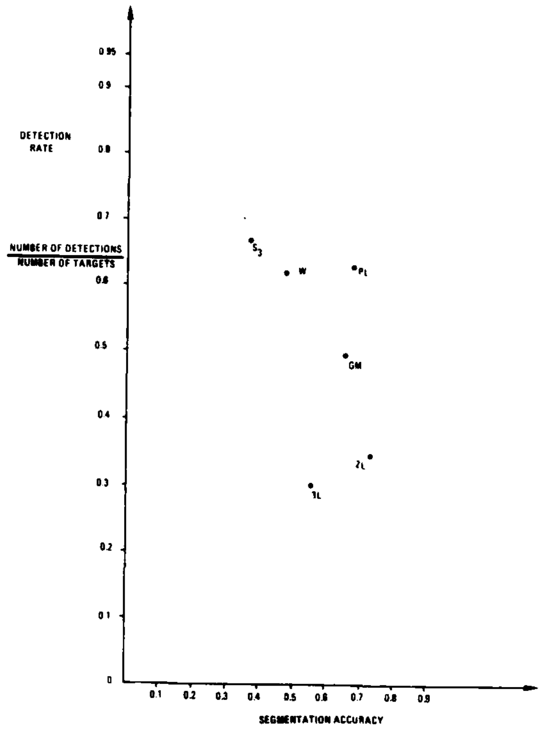


Figure 7. Detection Rate vs Segmentation Accuracy

SPOKE FILTER

Minor and Sklansky's [4] spoke filter is a generalization of the Hough circle detector [5]. The spoke filter uses an 8-spoke digital mask, which can be designed to detect either light or dark blobs, as illustrated in Figure 8. The arrows in the figure define a template. Each detected edge falling within the filter area is examined to determine if its strength exceeds a threshold and matches the direction of the corresponding template element. An 8-bit image size detection buffer is used to store edge matches. Each bit corresponds to one spoke direction. The n-th bit is set at buffer location (x,y) if at least one match is obtained along the n-th spoke when the filter is centered at (x,y). Upon completion of the spoke filtering operation, a 3 x 3 OR filter is convolved with the detection buffer. A detection is then considered obtained at position (x,y) if at least N bits of the (x,y) location of the buffer are set (typically N = 4).

Minor and Sklansky perform segmentation after detection is completed. The segmentor used is a modified version of the University of Maryland's Superslice [3,20,21] algorithm. Superslice views objects as being distinct from their surroundings by the presence of edges at their boundary. Superslice starts by choosing a threshold to segment the image into a set of connected components. It then extracts an edge map from the image. Finally, it measures the percent of each component's border which coincides with the edge map. An extracted blob is one which produces high edge-border coincidence. A number of thresholds are usually tried before good edge-border coincidence is obtained. Different thresholds may be required for extracting different objects in the same scene. Minor and Sklansky's version of Superslice also uses edge direction in the measure of coincidence.

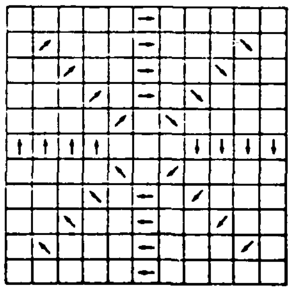


Figure 8a. Spoke Filter for Bright Objects on a Dark Background (From an Unpublished Version of [4])

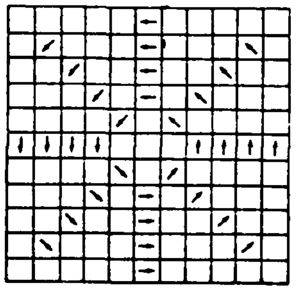


Figure 8b. Spoke Filter for Dark Objects on a Bright Background

A group at Hughes [6] briefly describes a method which tests each pixel in a region of interest for boundedness by edges in eight directions. A pixel is labeled to be interior to some boundary if it is bounded by edges in six of the eight directions. The labeled image is then thresholded to extract a target interior. If $\bar{Y} \leq \bar{X}$ then pixels with intensity greater than threshold C are selected, otherwise pixels with intensity less than C are chosen. Finally, a thinning and filling operation is used to remove small holes and smooth the segmented region. The Hughes report supplies no statistical model from which their thresholding equation is derived.

The author developed another algorithm based upon the basic spoke filtering philosophy. The algorithm uses sets of 3x3 pixel templates - with each template designed to match a section of a particular object's silhouette. The operation of the algorithm will be described by example.

First, suppose that an algorithm is to be designed to detect octagons of known polarity. A 4-bit image size buffer (initially zeroed) will be used to record detections. The algorithm will start by scanning across each of the rows of an image, applying a 3x3 vertical edge detector at each point visited. When a left blob edge is detected, its position will be noted (Figure 9a). The scan will continue until a right blob edge is detected (Figure 9b). At this time, a 0001_2 will be OR'd with the contents of the buffer at a location corresponding to the midpoint between detected edges (figure 9c). This process is repeated upon the image columns and in the two diagonal directions using the markers shown in Figure 10a.

At the completion of the scans in the four directions, we would expect to find a 1111_2 at buffer locations corresponding to centers of octagons in the original image. Since this may not always occur, due to the discrete nature of the image geometry, a 3×3 OR filter is convolved with the detection buffer.

Now suppose that we want to detect military vehicles. Instead of applying edge detectors in the four scan directions, feature detectors will be applied which correspond to shapes located around the sought vehicles' silhouettes. Also, the diagonal scan directions should not be exactly 45 degrees to image rows, but rather should be related to the average height to width ratio of targets (Figure 10b).

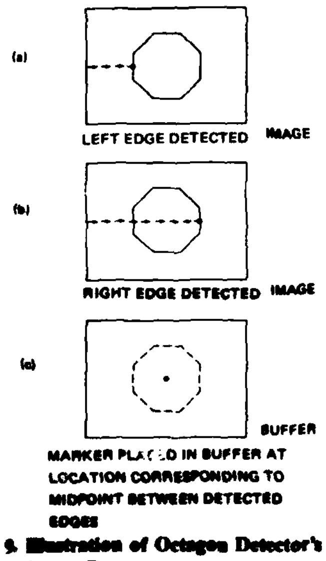

(a)
LEFT EDGE DETECTED
IMAGE
(b)
RIGHT EDGE DETECTED
IMAGE
(c)
BUFFER
MARKER PLACED IN BUFFER AT LOCATION CORRESPONDING TO MIDPOINT BETWEEN DETECTED EDGES

Figure 9. Illustration of Octagon Detector's Scan of an Image Row

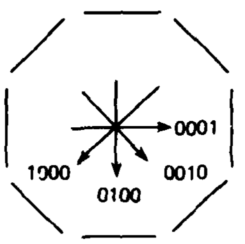


Figure 10a. Templates and Scanning Directions for Octagon Detector

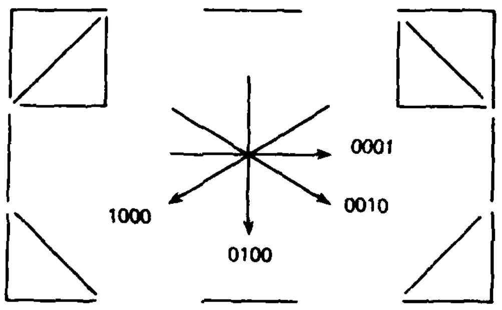


Figure 10b. Templates and Scanning Directions for Military Vehicle Detector (A Succession of 8 Features Represents a Particular Template)

Upon completion of the four scans, a spatial clustering algorithm is applied to detections, with cluster points replaced by their cluster centroid.

The algorithm performs very well as a detector as seen in Figure 6. Detection rate was 67 percent with 0.69 false alarms per target. The algorithm was also evaluated as a segmentor (Figure 7) with the distances between a centroid and the horizontal and vertical feature detections used to estimate target radii. Segmentation accuracy was not very good, with targets usually judged smaller than they actually were.

BORDER FOLLOWING/EDGE BASED SEGMENTATION

A blob outline containing no breaks is a closed curve separating the blob from its background. This curve may pass through the same pixel twice if the blob has a narrow neck near or at this pixel. A number of border following algorithms are described in the image processing literature, e.g., see [7-10]. They start by locating a prominent border point, and continue by visiting adjacent border points (edge elements) in sequence, eventually returning to the starting point. Difficulties arise when an object's boundary is not distinct over its entire length.

Schenker and Cooper [7,9] describe an elaborate algorithm which seeks the most probable blob boundary by conducting an exhaustive search over all boundaries. The "true boundary" is then viewed as the one which maximizes the joint likelihood of the observed data and a hypothesized boundary. Since an exhaustive search proves to be computationally prohibitive, suboptimal search algorithms are also proposed.

Perkins [15], while with General Motors Research Lab, developed an object detection/segmentation algorithm for use in a robot vision system. The algorithm is implemented by a sequence of simple operations. First, an edge map is obtained. Next, uniform intensity regions are extracted by expanding active edge regions, labeling the segmented uniform intensity regions, and then contracting the edge regions. The region with the most points along the image border is assumed to be the background. Finally, foreground region boundary points are visited; the locations of points which are roughly equidistant along a boundary are stored in an array. The performance of Perkins' algorithm as a FLIR target detector is not particularly good (Figure 6). The detection rate over the test data base was only 50 percent. The algorithm as implemented at GM does not use image polarity and, as a consequence, sometimes misses nearby targets, while indicating a detection half-way between them. The performance of the algorithm as a segmentor is excellent (Figure 6).

A simple border follower was programmed and tested at Westinghouse. A detection rate of 61 percent was obtained at a very high false alarm rate (Figure 7).

RELAXATION ALGORITHMS

Rosenfeld et al. [12, 19, 23], Kirby [13], Peleg [14], and others have viewed image segmentation as a graph labeling problem. The proposed approximate, iterative, parallel solutions are called "relaxation methods" which work as follows. First an initial set of labels is assigned to each image node based upon local image properties. The labels at each node are given weights between 0 and 1. This initial set of assignments may be ambiguous or incorrect in spots because of the imperfect nature of the local image measures or noise in the image. An iterative process using information obtained from neighbors is then used to improve the initial decisions. The weights at each node are simultaneously updated at each iteration (typically) based upon the weights at node neighbors during the previous iteration. This has the effect of altering the probabilities initially assigned to noise points to make them more consistent with their surround. The process is stopped when the labeling of all nodes seems to reach a steady state.

To apply 2-label relaxation to segmentation [23], a set of "light" and "dark" probabilities is assigned to image nodes based upon their gray levels. Probabilities at each node are then iteratively adjusted based upon probabilities at neighboring nodes, i.e., light reinforces light and dark reinforces dark. Eventually, the blob pixels should become uniformly light, and the background uniformly dark, so that segmentation is easy. The blob region is extracted by locating connected components. A 3-label scheme uses labels of "object", "background", and "clutter".

The relaxation algorithms tested performed rather poorly as target detectors (Figure 6).

PYRAMID APPROACHES

Let the size of an image be $2^n \times 2^n$ pixels. Reduced resolution versions can be linked with a pyramid data structure. The top (level 0) of the pyramid will be a size 1×1 image. The original $2^n \times 2^n$ image will be at the bottom of the pyramid. At level K will be an image of size $2^k \times 2^k$. A number of different schemes have been developed for detecting blobs with pyramids [24].

The pyramid linking scheme of Burt, Hong, and Rosenfeld [28] works as follows. A father-son relationship is defined between nodes of adjacent levels of the pyramid. Each node at level K is the father of a 2×2 array of "candidate son" nodes at level K + 1. Likewise, each node at level K is the son of four "candidate father" nodes at level K-1. At each iteration of the pyramid linking algorithm, a node is linked to the "most similar" of its four candidate fathers. A node is then assigned the average gray level of those sons linked to it. The process continues in this manner until a steady state is reached. The links then define trees, and the leaves of "light" trees are target segments. The particular pyramid linking algorithm tested did not perform well as a detector, yielding too many false alarms* (Figure 6).

MODE SEEKERS

There is a class of techniques in which a pixel is iteratively replaced by the average of a selected set of its neighbors. Narayanan and Rosenfeld [29] describe a method that chooses those neighbors for averaging which belong to the same histogram peak as the given pixel. The simplest version of this method chooses those neighbors higher up on the image histogram peak. This tends to move pixel gray levels toward their subpopulation modes. An improved version chooses a neighbor only if there is no significant concavity in the histogram between it and the given pixel. Ideally, upon termination, there will be a two-spiked histogram, with the lighter spike formed from target pixels. Target regions are then obtained by extracting connected components.

A local version of the algorithm performs processing over relatively small image windows in sequence, while the global version (named Global Superspike) uses the histogram for the entire image. The global version was tested since the images in the data base are only 128 × 128 pixels in size. Its detection rate was 88 percent (Figure 6).

EVALUATION

The mode seeker had the highest detection rate of any of the methods tested. Its segmentation accuracy fell in the 65 percent to 75 percent range, as did most of the other methods.

Spoke filters perform very well as detectors. But they have a low segmentation accuracy, and may need to be followed by a separate segmentor such as Superslice.

Double window filters work well as detectors. They can also be used for segmentation if only target extents are required. But preferably, they should be designed to seek out targets of particular sizes if range is available as an input. If targets of several sizes and orientations are sought, then a different filter is required for each. Each filter must vary in size down the image to correct for perspective. This has a multiplicative effect on hardware size and cost.

Simple border followers must be run at a high false alarm rate to yield a reasonable detection rate. They are well suited for segmentation, yielding estimates of target borders. If an entire image is available for processing, the border following can start at the most prominent border point and proceed from there. If an image is to be processed one line at a time (on-the-fly), then the topmost target point must be detected to initialize the follower. This second approach may have difficulty with non-convex targets but is easier to build into hardware.

Border followers and double window filters can be readily implemented in hardware. However, as noted above, proper implementation of the double window filter requires considerable computation power. The spoke filter in its purest form, as proposed by Minor and Sklansky, is not well suited for hardware

* Note: Another pyramid scheme was recently tested at the University of Maryland on this data with much better results.

implementation. It is computationally expensive requiring the repeated access of a large array of edges (or pixels) as the filter sweeps over the image. The algorithm can, however, be reformulated in a number of different ways to meet the requirements of particular computer architectures.

Relaxation algorithms are best implemented in a cellular array architecture having one processor per pixel. No one has ever built a target acquisition system using this architecture; to do so must therefore be viewed as a risky venture. The same holds true for pyramid approaches.

There are two difficulties with implementing the mode seeker. A FLIR image is typically 875 × 875. A mode seeker should be implemented on smaller image subregions, possibly horizontal strips (range zones). Secondly, the required post-processing step of extracting connected components is rather difficult to implement in real-time hardware. However, the image smoothing algorithm [29], upon which the mode seeker is based would be an excellent preprocessor for any detection algorithm. An analysis of its hardware implementability is not yet complete.

CONCLUSIONS

The following conclusions were drawn from the study.

- Mode seekers, double window filters, and spoke filters all show promise as targets detectors. Border followers and possibly mode seekers (if hardware implementation can be worked out) are fair segmentors.
- No existing computer architecture appears appropriate for the real-time implementation (on one or two circuit boards) of all detection/segmentation algorithms. A special architecture is required for the efficient implementation of each particular algorithm.

It was originally intended to test an artificial intelligence (AI) approach to improving algorithm performance. This appears to be much more difficult than initially thought. The investigation has not revealed any real-time target acquisition system which uses AI concepts. It is concluded that for the present efforts are better directed to the use of available information, such as range and feedback data. At a later time, an attempt can be made to incorporate a simple knowledge base and a limited reasoning ability. Several possible examples might be:

- Ground vehicles are likely to be located below the horizon.
- Targets of the same type often appear in groups, sometimes in moving columns on or near roads.
- Gathered intelligence indicating presence of different target types is often available.

A STRATEGY

A sound strategy is to precede detection by preprocessing filters designed to supress noise and "bring out" targets. The Narayan filter may be of use here. This should be followed by an initial detection algorithm which is cheap to implement in hardware, but which is not necessarily a "star" performer. This algorithm should be run at a low threshold (i.e., high false alarm rate) to make sure that most targets are detected. This should be followed by a more expensive detection/segmentation algorithm which will operate only on the initial detections. Performance should be improved by target tracking, frame-to-frame integration of extracted data, and decision smoothing.

ACKNOWLEDGEMENTS

This work was supported by the U.S. Army under the DARPA Image Understanding Program. The Army program monitor is Dr. George Jones of the Night Vision and Electro-Optical Laboratory, Ft. Belvoir, Virginia. The University of Maryland program director is Prof. Azriel Rosenfeld. The Westinghouse project manager is Dr. Glenn Tisdale. Prof. Rosenfeld and Dr. Tisdale reviewed a preliminary version of this manuscript and suggested improvements.

The FLIR data base was compiled by Rich Kroupa of Westinghouse. The Westinghouse border follower was developed by Arden Helland.

The experimental work at the University of Maryland was conducted by Les Kitchen, Ralph Hartley, and Cheng-ye Wang. The relaxation software was written by Russ Smith [25]. The mode seeking software was written by Krishnan Narayanan.

Dr. Marty Woolfson of Westinghouse suggested the double window probability density filter. Our evaluation of the hardware implementability of the various algorithms benefited from discussions with Westinghouse hardware designers Arden Helland, John Ferma, Jim Ross, and Rod Fletcher.

Special thanks goes to Dr. Walton Perkins and the staff of the General Motors Research Lab for testing our data base on their system.

REFERENCES

1. A.S. Politapoulos [Ford Aerospace], "An Algorithm for the Extraction of Target-like Objects in Cluttered FLIR Imagery," *AESS Newsletter*, Nov. 1980, pp. 23-31.
2. M. Burton and C. Benning [Texas Instruments], "A Comparison of Imaging Infrared Detection Algorithms", presented at SPIE 25th Annual Technical Symposium, Aug. 1981, San Diego.
3. (D.L. Milgram and A. Rosenfeld), "Algorithms and Hardware Technology for Image Recognition", Second Semi-Annual Report: (Nov. 1, 1976 - April 30, 1977), DARPA Order 3206, Contract DAA G53-76-0138, University of Maryland Computer Science Center, College Park, MD.
4. L.G. Minor and J. Sklansky, "Detection and Segmentation of Blobs in Infrared Images", *IEEE Trans Systems, Man, and Cyber*, March 1981, pp. 216-232.
5. C. Kimme, D. Ballard, and J. Sklansky, "Finding Circles by an Array of Accumulators", *Comm. ACM* 18, Feb. 1975, pp. 120-122.
6. R.L. Frey, C.D. Nealy, L.M. Rubin, and R.M. Wilcox, "ATAC Autocuer Modeling Analysis", Hughes Aircraft Co., Image Processing Lab., Culver City, CA, (for U.S. Army Night Vision and Electro-Optics Lab., Fort Belvoir, VA., report number FR-80-70-1325), January 1981.
7. H. Elliott, D.B. Cooper, and P. Symosek, "Implementation, Interpretation and Analysis of a Suboptimal Boundary Finding Algorithm", Proc. IEEE Pattern Rec. and Image Proc. Conf. (Chicago, Aug. 6-8, 1979), pp. 122-129.
8. A. Martelli, "An Application of Heuristic Search Methods To Edge and Contour Detection", *Comm. ACM*, 19, Feb. 1976, pp. 73-83.
9. P.S. Schenker, and D.B. Cooper, "Fast Adaptive Algorithms for Low-Level Scene Analysis: The Parallel Hierarchical Ripple Filter", SPIE Proc., Vol. 252, Smart Sensors II, Aug. 1980, pp. 113-123.
10. A. Rosenfield, "Extraction of Topological Information from Digital Images", Computer Science Dept. Tech. Report 547, Univ. of Maryland, College Park, June 1977.
11. Y. Yakimovsky, "Boundary and Object Detection in Real World Images", *J. ACM* 23, 1976, pp. 599-618.
12. A. Rosenfield, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations", *IEEE Trans. Systems, Man, and Cyber.* 6, June 1978, pp.420-433.

13. R. Kirby, "A product rule Relaxation Method", *Computer Graphics and Image Proc. 13,* June 1980, pp. 159-189.
14. S. Peleg, "A New Probabalistic Relaxation Scheme", *IEEE Trans. Pattern Analy Mach. Int.* 2, July 1980 pp. 362-368.
15. W.A. Perkins, "Area Segmentation of Images Using Edge Points", *IEEE Trans. Pattern Analysis Mach Int.* 2, Jan. 1980, pp. 8-15.
16. W.A. Perkins, "Simplified Model-Based Part Locator", Proc. Fifth Intl. Conf. on Pattern Recognition (Miami Beach, FL, Dec. 1-4, 1980), pp. 260-263.
17. B. J. Schachter and G. E. Tisdale, "Evaluation and Real-Time Implementation of Image Understanding Algorithms", Proc. Image Understanding Workshop, April 1981, pp. 178-183.
18. R.V. Hogg and A.T. Craig, *Introduction to Mathematical Statistics,* MacMillian, 1970.
19. A.J. Danker and A. Rosenfeld, "Blob Detection by Relaxation", *IEEE Trans. Systems, and Man, and Cyber.* 3, Jan. 1981, pp. 79-92.
20. D. Milgram, "Region Extraction Using Convergent Evidence", Proc. Image Understanding Workshop, April 1977, pp. 58-64.
21. D. Milgram, "Progress Report on Segmentation Using Convergent Evidence", Proc. Image Understanding Workshop, Oct. 1977, pp. 104-108.
22. T. Silberberg, S. Peleg, and A. Rosenfeld, "Multiresolution Pixel Linking for Image Smoothing and Segmentation", in Proc. Image Understanding Workshop, April 1981, pp. 33-38.
23. A Rösenfeld, A. Danker, and C. R. Dyer, "Blob Extraction by Relaxation", Proc. Image Understanding Workshop, April 1979, pp. 61-65.
24. A. Rosenfeld, "Some Uses of Pyramids in Image Processing and Segmentation", Proc. Image Understanding Workshop, April 1980, pp. 112-115.
25. R.C. Smith, "A General Purpose Software Package for Array Relaxation", University of Maryland, Computer Science Center, TR-839, Dec. 1979, College Park, MD.
26. T. Silberger, S. Peleg, and A. Rosenfeld, "Multiresolution Pixel Linking for Image Smoothing and Segmentation", University of Maryland Computer Science Center, TR-977, Nov. 1980, College Park, MD.
27. K.A. Narayanan, S. Peleg, A. Rosenfeld, and T. Silberberg, "Iterative Image Smoothing and Segmentation by Weighted Pyramid Linking", University of Maryland Computer Science Center, TR-989, Dec. 1980, College Park, MD.
28. P. Burt, T.H. Hong, and A. Rosenfeld, "Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchical Computation," University of Maryland Computer Science Center, TR-927, Aug. 1980, College Park, MD.
29. K. A. Narayanan and A. Rosenfeld, "Image Smoothing by Local Use of Global Information", University of Maryland, Computer Science Center, TR-1006, Feb. 1980, College Park, MD.

AD-P000 112

EXTRACTING COMPACT OBJECTS USING LINKED PYRAMIDS

T. H. Hong
M. Shneier

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

Compact objects of arbitrary size are extracted from images using a combination of three-pyramid-based representations of image features. A gray-scale linked pyramid is used to smooth the image into uniform regions. A "surroundedness" pyramid is used to identify regions of interest, and a linked edge pyramid is used to delimit the boundaries of the compact objects.

1. Introduction

Many image processing tasks require the extraction of objects from a background. Most notable among these is target detection. In many cases there is some a priori knowledge about the shapes and sizes of the objects, which could aid in their extraction. Unfortunately, it has not normally been possible to extract objects that have the right size and shape without extracting other, unwanted objects as well. Removing the unwanted objects then requires another stage of processing, which can be very complicated if the desired objects are embedded in background clutter.

This paper presents a pyramid-based method of extracting compact objects that is able to apply knowledge about the size and shape of an object directly to the segmentation process, to avoid extracting unwanted regions. The method provides solutions to a group of problems, including object detection, edge completion, and region filling. It makes use of both gray-scale and edge information. In addition, it computes a surroundedness measure for each pixel, representing the degree to which that pixel is locally surrounded by edges. All three sets of information - gray level, edge magnitude and direction, and surroundedness - are represented in pyramid structures, and it is the interaction between the different types of information at each level of each pyramid that leads to the final segmentation. The representations are, themselves, built on one another. A gray-level pyramid is used to construct an edge pyramid, which is in turn used to construct a surroundedness pyramid.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76-C-0138 (DARPA Order 3206) is gratefully adknowledged, as is the help of Clara Robertson in preparing this paper.

The gray-scale pyramid is used to segment the original image into smooth regions that are not necessarily connected (Section 2.1). It is common for an object to belong entirely to one of these regions, but the algorithm does not require this to be the case. The edge pyramid (Section 2.2) is used in two ways. The edges indicate parts of the image that could be individual objects, enabling the objects to be separated from the regions extracted by the gray-level pyramid. The edges also serve as the basis for constructing the pyramid of surroundedness scores (Section 2.3).

The surroundedness scores are used to find starting points for a combined region-growing and region-splitting process. The growth of the region is controlled by the gray-level pyramid, and the region is pruned by the edge pyramid. In this sense, the method is analogous to the "superslice" algorithm (Milgram, 1979) and to the relaxation method of Danker and Rosenfeld (1979). One of the notable features of the method is that the region does not "leak" through holes in its border. This is partly because of the pyramid's tendency to bridge small gaps as the resolution decreases from level to level.

A previous use of a pyramid process for extracting compact objects (Shneier, 1979) made use only of gray values and a compactness measure. For each compact region that was discovered, a threshold was computed and applied in a square region of the original image to extract the object. The current method does not use a threshold to extract the regions, but makes use of edge information to determine the shapes and sizes of the regions.

The process of constructing the pyramids is described in Section 2, and the succeeding section describes how each pyramid is used to arrive at the final result. Examples are given of applying the system to a set of images, and the results are compared with those obtained in a recent segmentation study (Hartley et al., 1981).

In the following sections, the pixels at each level of the various pyramids play two roles. They are points in an image at some level of a pyramid, and are also nodes in the tree structure defined by the links between levels in the pyramids. Both names will be used interchangeably.

2. Constructing the pyramids

2.1 Gray-scale pyramid

A gray-scale pyramid is a sequence of square images, each a lower-resolution version of its predecessor. The kind of pyramid used in this work is the linked structure defined by Burt et al., (1980). It is constructed as follows.

Each level is formed by summarizing a 4 by 4 neighborhood in the preceding level. The neighborhoods are overlapped fifty percent vertically and horizontally so that each pixel has four "fathers" at the next level, and sixteen "sons" at the previous level. The average or the median of the sixteen sons can be used as the summarizing value for their father. In the implementation, the average value was used.

The entire pyramid is constructed in this way, up to the level at which there are only four pixels. There follows an iterated linking process in which each node is linked to that one of its four fathers whose gray value is most similar to its own. A father can thus have up to sixteen sons, while a son can have only one father. After the links have been established, each node recomputes its gray value based only on the values of the sons linked to it. This process is iterated, and usually stabilizes after a few iterations.

At this stage, each pixel at the bottom level of the pyramid (the original image) is linked through some sequence of ancestors to one of the four pixels in the topmost (2 by 2) level of the pyramid. Each topmost node thus represents some region in the original image, which can be extracted by following links down the pyramid. If the values of pixels in the original image are replaced by the corresponding values of their ancestors, a segmentation of the image into at most four regions is obtained. It is not necessary that these regions be connected.

The segmentation defined by this procedure is not necessarily in terms of objects and background. Indeed, for the image in Figure 1, the chromosomes are extracted as one component, while the background is segmented into three components of slightly different average gray-value. Often, a desired region belongs to one of the four components, but is lost among the other parts of the image that link to the same component. As an example, notice that one of the small chromosomes in Figure 1b disappears entirely. The procedure defined in this paper is largely concerned with isolating individual parts of the four components into separate objects, although it is also able to merge parts from different components into a single object. The process relies on edge and surroundedness information to find the subcomponents to be extracted.

Figure 1 shows an image and the results of iterating the gray-level linking process. The resulting preliminary segmentation forms the input to the rest of the procedure.

2.2 Edge Pyramid

The edge pyramid is constructed by first building a gray-level pyramid and then applying an edge operator at each level to produce an edge pyramid (Hong et al., 1981). The gray-level pyramid used for extracting edges was based on non-overlapped 2 by 2 blocks, and the values at each level were defined as the medians rather than the means of the values in the blocks at the level below. This reduces the amount of blurring and distortion of the edges (Tanimoto, 1976).

The edge operator that was used is one that scored highest in the edge evaluation tests of Kitchen and Rosenfeld (1981). It is the three-level template operator (Abdou and Pratt, 1979) which uses eight direction masks, e.g.

$$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \quad \text{and} \quad \begin{matrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{matrix}$$

The edge detection is followed by a non-maximum suppression stage. A 3 by 3 window is placed around each edge point. The direction of the edge is used to find the two edge points to use for non-maximum suppression. If the edge point has a magnitude greater than both points, and a direction difference of less than 45 degrees, it survives; otherwise, it is deleted. Figure 2a shows an edge pyramid constructed from the chromosome image.

Edges, too, are linked together between levels. For linking purposes, the pyramid is assumed to map each point to a 4 by 4 region in the level below. Once again, each son has four potential fathers and each father has sixteen sons. Linking proceeds bottom-up. Each son compares his direction with those of his four fathers, and chooses the father whose direction is most compatible. If the difference in directions is less than some threshold (here 46 degrees), the son is linked to the father. Otherwise, the son becomes the root of a tree. Ties are broken by choosing the first father that satisfies the criteria. The direction of a son is updated to become the average of the son's direction and the father's direction, but the process is not iterated.

2.3 Surroundedness pyramid

The edges at each level of the edge pyramid are directed in such a way that the brighter side of the edge is to its right. This information could be used by itself to prune the gray-level pyramid by demanding that the gray levels at positions corresponding to opposite sides of an edge obey this constraint. Such a process would not necessarily lead to a segmentation into compact objects. It is first necessary to identify the edges that bound compact objects, and to ignore all other edges. A procedure for finding such edges was described in Hong et al. (1981).

In the current system, however, the aim is to extract the interiors of compact regions. The process is applied at each level of the pyramid, and compact objects of different sizes are identified at different levels. There are two stages involved in finding compact regions from edge information.

First, the skeleton of a region is found by looking at 5 by 5 neighborhoods of each point. There is no need to look further than two points on either side of a pixel, because, if there are no edges within this distance, the object will become more compact at the next higher level of the pyramid, where the process is applied as well. The aim is to find interior points of a region that are surrounded by edge points with compatible directions.

Let x be the central point in a 5 by 5 neighborhood (Figure 3). The remaining points in the neighborhood are divided into three classes. The points marked A are the immediate neighbors of x, while those marked B and C are more distant from x. The numbers associated with each point are their chain code orientations in units of 45 degrees. Finding the skeleton proceeds as follows.

If the edge magnitude of x is not zero, ignore this point, because x is not interior.

If the magnitude is zero, check the neighbors of x:

1. For each type A neighbor of x whose edge magnitude is not zero, the edge direction of A is allowed to differ from its chain-code direction by no more than some threshold (here 23 degrees). For example, the edge direction of the point immediately East of x must lie between -23 degrees and +23 degrees, while the edge direction of the point North-East of x must lie between 23 degrees and 45 degrees. That is, the edge directions should be consistent with the edges of a closed region. If this condition is met, the score for the particular direction from x is set to 1. The score is a measure of how central the point is, i.e., of its membership in the skeleton of the region. For each point, there are eight slots for scores, corresponding to eight directions. A perfect border around x would result in all eight slots being set to 1. Note that more than one point in the 5 by 5 neighborhood can set the same slot value.

2. If the magnitude of a type A neighbor of x is zero, the neighboring type B point is examined as above. If its edge direction is compatible with its grid position, the score for x is set to 1.

3. For all type C points whose edge magnitude is not zero, the corresponding direction slot for x is set to 1 if the direction of the point is within 23 degrees of the chain-code position. For type C points, however, the chain-code direction is calculated at 45 * chain-code number + 23, because type C points are offset an extra 23 degrees from x.

Notice that all the type A and type C points contribute to the score for x, but type B points only contribute if the neighboring type A point is not an edge point. This is because closer edges are assumed to block the effects of edges that are more distant, and hence less likely to belong to the same object. This is particularly important at high levels of the pyramid where the objects are very close together.

When the scoring process has been applied to each 5 by 5 neighborhood at each level in the pyramid, the second stage of finding compact regions is performed. The purpose of the second stage is to propagate the score of the skeleton out to the borders of the region. For the second stage, the score is computed as the sum of the slot values. A threshold is applied to decide what score values are considered to constitute valid skeleton points (here a score of 5 out of a possible 8 was used). For each such point the following procedure is performed.

1. For all type A or C points whose edge magnitudes are not zero and whose edge directions are compatible (as in the previous step), assign a new score which is the maximum of the current score and the sum of the slot values for x (the skeleton point).

2. For type A points whose edge magnitude is zero, check the corresponding type B point. If its magnitude is not zero and its direction is compatible, assign a new score to both the type A point and the type B point. In each case the score is the maximum of the score for x and the current score for the point.

When both steps of the process have been completed, each compact region will contain a set of high scores, as will the edge points surrounding the region (Figure 2b). These points define the extent of the region at the particular level in the pyramid. To extract the corresponding region in the original image requires the use of both the gray-level and the edge pyramids. The particular scoring function used does not have any special significance, and it is likely that other functions would perform equally well.

Note that no thresholding was used to discard edges with very low magnitudes. It is sometimes useful to keep only the strong edges, and so avoid extracting objects with very low contrast that are invisible to the human eye. To a large extent the loss of resolution at higher levels of the pyramid achieves this automatically, but it is true that at low levels in the pyramid a lot of small noise regions might be extracted. Examples of the improved performance resulting from thresholding the edge magnitudes are shown in Section 4.

The surroundedness pyramid has no links between the levels. As a result, compact objects can be detected at more than one level of the pyramid. In previous work (Hong et al., 1981) links were established, and the object was detected at the highest level at which it was well defined. Such a process would probably work for the current pyramid structure as well.

3. Extracting the compact regions

The most obvious way of extracting compact regions from a given level in the surroundedness pyramid is first to find all points that have a high surroundedness score. These points can then be projected down to the base level by finding the corresponding points in the gray-level pyramid and following their links. Unfortunately, this simple process results in regions that are displaced, misshapen, and which have holes and protrusions that do not appear in the original objects.

There are a number of reasons for these imperfections, analysis of which leads to a more complex extraction process, but one that produces regions that are much closer to the actual shapes of the objects. The flaws can arise from a poor initial segmentation in the gray-level pyramid and from displaced or missing edges in the edge pyramid. Poor edge data also lead to incorrect surroundedness information, and this also must be improved.

The compact region developed from the edge information can be incorrect for two reasons. First, the edges could be misplaced due to the averaging in the pyramid process and the non-maximum suppression applied at each level. Second, there may be missing or noisy edges. To correct the placement of the points, use is made of information from the gray-level pyramid. If the object is known to have a particular color, then all points with that color that are in the compact region (i.e. have a high surroundedness score) can be called object points, and the rest can be ignored. Alternatively, if the object is known to be, say, the brightest region in the image, the gray value of the brightest node of the 2 by 2 level in the gray-level pyramid can be projected down and intersected with the compact points to give a more accurate compact region. Usually, however, the relative brightness of the object is not known, or the object may have more than one color, so that a more conservative approach has to be taken. This involves a local process to identify the set of gray values that occur most commonly in the interior of the compact region. These values are taken as representing the object, and neighboring points with the same gray values are added to the starting set to give a new compact region whose position is more accurate because it is derived both from edge- and region-based properties.

To correct for missing edges, the structure of the edge pyramid is used. As the resolution of the pyramid increases towards its base, the positions of the edges become more and more accurate, but the gaps become larger and larger. By fitting lines through existing edge points in a top-down process, the gaps can be filled in relatively cheaply, and should approximate the actual contours of the boundary more and more closely as the resolution increases.

Another problem that arises from using edge information is that holes can appear inside object regions because of noise in the image. For most applications that were implemented, no edge magnitude thresholding was performed, and for all applications, no thresholding was performed above the base level of the pyramid. As a result, edge points with very low magnitude often appear in the interior of objects. Again, by taking advantage of the pyramid process, it is possible to remove these edges and so ensure that holes do not appear inside the objects. A characteristic of noisy edges is that they do not survive as the resolution of the image is reduced at successive pyramid levels. By examining the sons of interior points and deleting those that are edge points, the interior of the region can be cleaned up. Of course, it is possible for holes that are real features to be eliminated in this way, and it is likely that edges with magnitudes above some threshold should be retained.

The final difficulty of using naive projection to find the compact regions is that the objects that are found have misshapen boundaries. In some places, the boundary might extend into the background, while in others it might not extend out to the actual border. It is even possible for the simple projection process to give rise to disjoint regions at the base of the pyramid. This problem is overcome by projecting down the gray values level by level, and using the edges at successive levels to delimit the borders.

The process of extracting regions involves a simultaneous addition and deletion of nodes in the gray-level pyramid, guided by the edge and surroundedness pyramids. Nodes are added if they are on the interior side of an edge and adjacent to a compact point. They are deleted if they are on the outside of an edge belonging to the compact object. The additions and deletions are performed top-down at each level of the pyramid below the level at which the compact object was discovered. The result is a region whose outline closely follows the edge bounding the object, and which is tolerant of gaps in the edge information. This is similar to the process described by Strong and Rosenfeld (1973), but occurs vertically across levels of the pyramid, instead of horizontally within a level. In more detail, the process is as follows.

1. Project the gray values from the top (2 by 2) level of the pyramid down to the level at which the compact object was discovered (i.e., the level at which it received an above-threshold surroundedness score). Call this level L.

2. Choose points to be considered as part of the object from among the points belonging to the compact region as follows. For every point x in level L that is an interior point (i.e., has a high score and is not an edge point), examine the surrounding 5 by 5 window. If x has the same value as the majority of its neighbors, then x is considered a valid object point. This ensures that points that have gray values that belong to the background, or have a mixture of the region and background colors, are not included.

3. Expand the set of points belonging to the compact object by again looking at 5 by 5 neighborhoods, this time for all points x at level L, regardless of whether they are interior points or not. If x has neighbors in the 5 by 5 region that were chosen as object points in the previous step, then x is marked as an object point if x has the same gray value as one of those points. This compensates for shifts in the edge positions due to the pyramid process and the non-maximum suppression.

4. Project the nodes in the enlarged compact region down one level in the gray-level pyramid, to level L-1.

5. Examine interior points of the compact region in the edge pyramid at level L. If any of the central four sons of an interior point are edge points, delete them. This cleans out noisy edge points in the interior of the object at level L-1.

6. At level L-1, expand the compact region by examining edge points that link all the way to level L. If these edge points have interior neighbors that are not part of the region, add them in regardless of their gray value. This expands the region to fit the boundary at the current level.

7. Fit lines through the edge points of 7 by 7 neighborhoods at level L-1 (see below). Delete points that lie outside these lines if they are part of the compact region. This ensures that the region does not grow outside the edge boundary, and prevents leaks where no edges exist.

8. Repeat steps 4 - 7 for levels L-2, L-3,... until the bottom of the pyramid is reached. At this stage, the compact region has been extracted.

Lines are fitted to edge points below level L to fill in gaps in the edges. For every edge point x that links to the border of an object at level L, a set of points (e.g., those marked a in Figure 4 and their rotations) is examined if x satisfies the following conditions.

1. x must not be surrounded by interior points. This assumes that the objects do not have holes in them, and can be relaxed if necessary.

2. There is no edge parallel to x in the area marked by a's in Figure 4. This is because the parallel edge will prune the region and, since edge magnitudes were not used, the outermost edge is considered the real edge at the current level.

If both conditions are satisfied, all the points marked a are pruned. In the implementation, points were only deleted if they did not link to <u>any</u> compact object. This was because all compact regions were being extracted simultaneously, and it was possible for points from a different object to appear in the neighborhood, especially at high levels in the pyramid.

The reason for projecting the values from the 2 by 2 level to level L and using the set of points that have the most common gray values is to alleviate effects that the edge construction process has on the position of edges. Assume that an object is represented mostly by a single gray value in the original image, and that this consistency is preserved at all levels of the pyramid. Then, so long as the edges do not shift too far, the intersection of the compact region and the set of points with the most common gray values is a good seed for growing the region. Adding in points that are immediate neighbors of the seed points and that have the same gray values ensures that the region is shifted appropriately. It does not matter too much if the corresponding region at the bottom of the pyramid is too large, because the pruning that takes place at lower levels will make sure that the region stays within the boundaries defined by the edges. Note that the shift in the edges is greatest at the top of the pyramid, and becomes less and less as the base level (the original image) is approached. Because of the links between levels, the shifting is not particularly important. Every projection follows the links, both in the gray-level and the edge pyramids, so that the size and position of the region converges to the true size and position of the corresponding object as the base of the pyramid is approached.

Note that no threshold was applied to the edge magnitudes, so that many weak edges remain at each level. Most of these do not form links to the next level, or, at least do not survive as the size of the region that contains them shrinks. The noise-cleaning step examines interior (i.e. non-edge) points at one level and deletes any of their central 2 by 2 sons that are edge points. This step can sometimes cause interior detail to be lost. For example, in the image of Figure 5, the central dark region is filled in. Usually, however, the process ensures that there are not holes in the final object.

The step of expanding the region to conform with the edge data accounts both for the fact that the gray-level pyramid might not match the edge pyramid exactly and for the possibility that the gray values of the object might not be uniform. Many objects exhibit a smooth transition with the background. By expanding the region, guided by the edges, it is possible to account for variations in gray values.

Region splitting is applied for similar reasons. If there is no change in gray values between the object and the background in the gray-level pyramid, then many points outside the object will be linked to nodes that are interior nodes at a higher level in the pyramid. When the gray values are so similar, it often happens that no edges are found at the corresponding positions on the edge pyramid. By the nature of the pyramid, however, a missing segment becomes smaller and smaller as the height of the pyramid increases. By interpolating across small breaks at each level, a close approximation to the actual boundary can be obtained. This interpolation is done by fitting lines through the edges at each level. All nodes that lie outside these lines are pruned, while those inside are added to the object. As the resolution increases down the pyramid, the fitting process approximates the object boundary more and more closely.

4. Examples

The procedure was applied to a set of FLIR images and to a picture of a number of chromosomes of varying sizes. On the whole, the results were very satisfactory, although the method is less successful when the objects are so small as to appear only in the original image. In these cases, there is no smoothing effect from the pyramids and, because there is no thresholding of the edge magnitudes, a number of small noise regions are extracted together with the desired regions.

Figure 6 shows a very clean example of the system's abilities. The original image consists of a number of chromosomes against a dark background. There are no objects so small as to be visible only at the full-resolution level of the pyramid, and the objects are spread out by size across the next two levels. Thus, the smaller chromosomes appear in Figure 6a, while the larger chromosomes appear in Figure 6b. The larger chromosomes are also extracted in Figure 6a because their surroundedness score is high enough at this level. The process mentioned earlier of choosing the best level at which to extract an object would enable the larger chromosomes to be extracted only at the higher level.

It should be realized that each chromosome is extracted individually, even though the gray-level pyramid links them into a single top-level node. The chromosomes are extracted cleanly, despite the gaps in edge information evident in the edge images (Figure 2), and despite the fact that one small chromosome is totally lost in the background of the gray-level pyramid.

Figure 7 shows an example of the expansion of the gray-level pyramid region to fit the edge image. In the upper left image, the original gray-scale tank merges fairly smoothly with the background. This results in an original compact region smaller than the actual tank (bottom left). The compact object was actually found at the 8 by 8 level of the pyramid, and the bottom right image shows the results of adding in points on the inside of the edge data at the 16 by 16, 32 by 32, and 64 by 64 levels of the pyramid. The result is a region whose shape is a close approximation to the shape of the actual object.

Figure 8 shows an example where parts of the region outside the object are discarded by the pruning step. A node was removed because it was on the wrong side the region boundary, resulting in a more accurate outline. In fact, such pruning happens in almost all the images.

Figure 9 shows what happens when the objects being sought are too small. If an object is not large enough to be represented at a level above the original image, the only filtering taking place is due to the surroundedness scoring. It is possible for a single noise point to give rise to a compact region, and this would be detected in addition to any legitimate targets. Noise cleaning at this level eliminates many of the detected objects, but can remove the desired objects as well. By thresholding the edge magnitudes, however, a much better result can be obtained. A similar improvement could be expected if the surroundedness scoring took the edge magnitudes into account. Even without any thresholding, the number of regions detected is still less than that for the gray-level linking based segmentation. On these same images objects that are large enough to survive even to the first level above the original image are detected with almost no background clutter. Figures 10a to 10s show the results obtained when the edge magnitude is thresholded (at 15). Figures 11a to 11i, 12a to 12f, and 13a to 13p show the objects extracted at successively higher levels without edge magnitude thresholding.

In the segmentation study of Hartley <u>et al</u>. (1981), the gray-level linking method of segmentation performed reasonably well, except for the detection of a large number of unwanted objects (false alarms). The current method, being based on the gray-level linking process, is guaranteed to do no worse than that method. In fact, the results show that the method significantly reduces the number of false alarms, and often eliminates them entirely. The method can also be tuned to detect objects of a particular range of sizes, and does so with no extra processing. If the method were to be ranked using the scoring function of Hartley <u>et al</u>., it would rank ahead of all the methods they tested. (Table I). Note that most of the images in the study had to be sampled down to 64 by 64 pixels because that is the largest size the program can handle. For those images for which sampling was not necessary (11-30), the method performed better than the others. Overall, the method was as good at detecting targets as the best method in that study, but had a lower false alarm rate, and no extra detections. The method would probably perform even better if it were re-implemented to handle full-resolution images.

5. Discussion and Conclusions

A method has been presented that extracts compact objects from images. The method uses three kinds of pyramid-based representations. The first is a gray-level pyramid, with links between points at successive levels. The second is a pyramid of edge information for each level, and the third is a surroundedness pyramid that reflects the compactness of regions at each level.

The results of applying the method to a number of images indicate that it is successful in extracting compact objects so long as they are large enough to survive at least to the second level of the pyramid. The extracted objects have borders that closely follow the outlines in the original scene, as found by the edge detector, and very few extraneous regions are usually detected. Even in the cases in which the objects are very small, they are still usually extracted, although a number of unwanted regions might also be extracted. By thresholding the edge magnitudes of the original image, most of the unwanted regions can be discarded, leaving only the compact objects. It can also be seen that the process extracts only compact regions. For example, the road in Figure 14 is not extracted, because it is elongated rather than compact.

Levine (1980) discussed a pyramid-based algorithm for region analysis that is related to the approach presented in this paper. He made use of three color pyramids, a texture pyramid, and an edge pyramid. None of the pyramids were constructed using overlapping regions, and the edge pyramid was formed by ORing 4 by 4 regions of an original edge image to produce the successive levels. The aim of the research was not to extract objects with particular shapes, but to segment a scene into regions. Processing involved finding points as far away from the borders of regions as possible, by finding the levels in the edge pyramid abovae which a set of edges disappeared. These points then served as seeds for growing regions by projection in the pyramids. At each level, the boundaries between regions were refined by a close examination of the neighboring points. When the final projection was completed, a clean-up process was used to merge small regions with adjacent larger regions. The method proposed in this paper makes more use of local gray values in the analysis, and does not need to perform any postprocessing of the image.

Earlier work has also concerned the problem of filling in regions from broken edge information. Strong and Rosenfeld (1973) describe an iterative procedure that simultaneously grows regions and fills in gaps in the borders. The method described here has advantages in that the speed with which regions can be filled in is significantly greater in the pyramid, as is the distance over which gaps in the edges can be bridged.

Danker and Rosenfeld (1979) examined the use of pyramids to speed up the propagation of edge and region labels in their relaxation scheme for extracting regions, but their results were inconclusive. Given the ability to perform operations in parallel, the current method can be made very efficient. The pyramids are all constructed in one pass, although the gray-value pyramid linking process is iterated. Later processing involves a single pass through the pyramid, starting at the level at which the compact object is found, and ending at the level of the original image. All processing within and across levels is local in nature, so that the potential exists for real-time implementation of the algorithm. To make the results comparable with the study of Hartley et al., the gray-level linking process was iterated. It is not clear that this is necessary because the process does not depend on having regions with uniform colors.

It would be of interest to extend this work by devising scoring functions to detect elongated objects, for example, or objects of arbitrary shape. With a small set of primitive shape recognizers it would be possible to build a powerful system that could selectively extract objects having a wide variety of shapes.

REFERENCES

1. I. E. Abdou and W. K. Pratt, Quantitative design and evaluation of enhancement/thresholding edge detectors. Proceedings IEEE 67, 1979, 753-767.

2. P. Burt, T. H. Hong, and A. Rosenfeld, Segmentation and estimation of image region properties through cooperative hierarchical computation. IEEE Transactions Systems, Man, Cybernetics 11, 1981, 802-809.

3. A. Danker and A. Rosenfeld, Blob detection by relaxation. IEEE Transactions Pattern Analysis Machine Intelligence 3, 1981, 79-92.

4. R. L. Hartley, L. J. Kitchen, C. Y. Wang, and A. Rosenfeld, A comparative study of segmentation algorithms for FLIR images. TR-1104, Computer Science Center, University of Maryland, College Park, MD, September 1981.

5. T. H. Hong, M. Shneier, and A. Rosenfeld, Border extraction using linked edged pyramids. TR-1080, Computer Science Center, University of Maryland, College Park, MD, July 1981.

6. L. J. Kitchen and A. Rosenfeld, Edge evaluation using local edge coherence. IEEE Transactions Systems, Man, Cybernetics 11, 1981, 597-605.

7. M. D. Levine, Region analysis using a pyramid data structure. In Structured Computer Vision (S. Tanimoto and A. Klinger, eds.) Academic Press, New York, 1980, 57-100.

8. D. L. Milgram, Region extraction using convergent evidence. Computer Graphics Image Processing 11, 1979, 1-12.

9. M. Shneier, Using pyramids to define local thresholds for blob detection. TR-808, Computer Science Center, University of Maryland, College Park, MD, September 1979.

10. J. P. Strong III and A. Rosenfeld, A region coloring technique for scene analysis. Comm. ACM 16, 1973, 237-246.

11. S. L. Tanimoto, Pictorial feature distortion in a pyramid. Computer Graphics Image Processing 5, 1976, 330-352.

Images	Targets	Method	Correctly detected	Extra detections	False alarms	Segmentation accuracy
2-10 (Navy, China Lake)	8	2-class relaxation	0	0	43	-
		3-class relaxation	2	0	67	0.70
		Pyramid linking	0	0	145	-
		Superspike	3	0	77	0.51
		Surroundedness pyramid	4	0	32	0.60
11-30 (NVL data)	80		40	0	92	0.73
			20	8	92	0.49
			72	32	392	0.67
			76	24	60	0.64
			76	0	16	0.73
31-36 (Air Force, TASVAL)	6		2	0	9	0.74
			3	1	27	0.73
			3	2	100	0.57
			6	1	63	0.60
			5	0	11	0.70
55-70 (NVL flight test)	32		2	0	6	0.67
			13	1	19	0.65
			4	0	38	0.80
			26	1	2	0.73
			26	0	7	0.60
Overall	126		44	0	150	0.73
			38	10	205	0.58
			79	34	675	0.68
			111	26	202	0.66
			111	0	67	0.69

Table I. Summary of results for the comparative segmentation study.

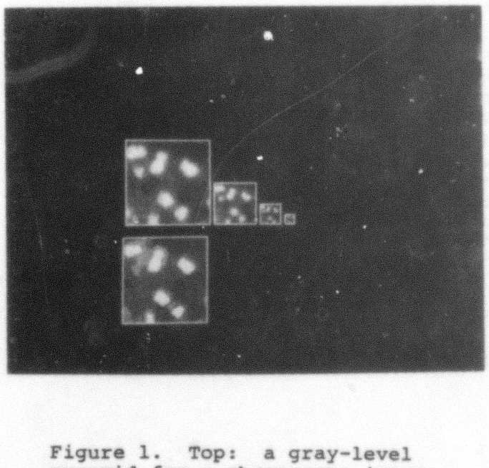


Figure 1. Top: a gray-level pyramid for a chromosome image. Bottom: the results of iterating the gray-level linking process (10 iterations).

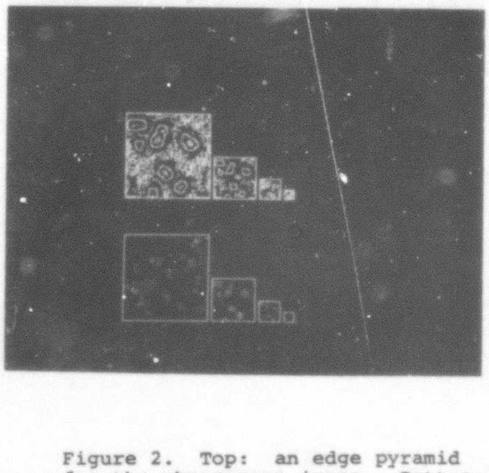


Figure 2. Top: an edge pyramid for the chromosome image. Bottom: a surroundedness pyramid for the chromosome image.

B3	C2	B2	C1	B1
C3	A3	A2	A1	C0
B4	A4	x	A0	B0
C4	A5	A6	A7	C7
B5	C5	B6	C6	B7

Figure 3. The 5 by 5 neighborhood for computing surroundedness scores. The numbers denote chain-code directions.

				a		
				a	a	
				a	a	a
			←x	a	a	a
				a	a	a
				a	a	
				a		

a	a	a	x↗			
a	a	a	a			
a	a	a	a			
a	a	a	a			

Figure 4. The 7 by 7 neighborhoods used to fit lines through edge points. The arrow indicates the direction of the edge point x, and the a's indicate the region that is examined. Rotations of these patterns are used for other edge directions.

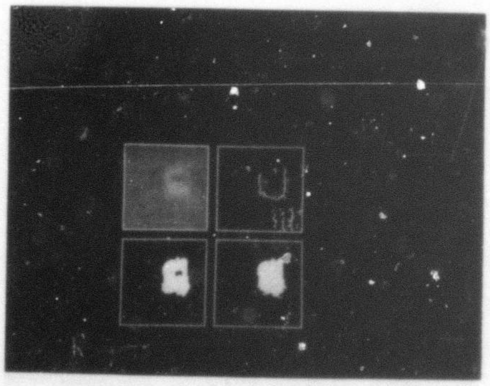


Figure 5. Top left: the original FLIR image of an armored personnel carrier. Top right: the edge image projected down from the level at which the compact object was found (8 by 8). Bottom left: the compact object found at level 3 (8 by 8) without deleting interior edges. Bottom right: the result of applying the whole process to the image. The hole in the middle has been filled in.

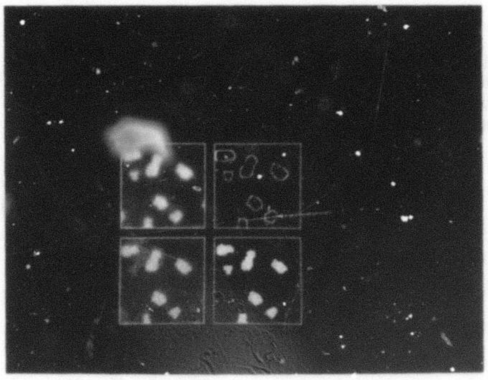
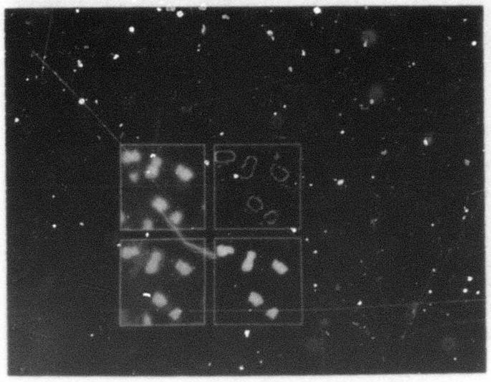


Figure 6. a) The chromosomes extracted at level 1 (32 by 32), the first level above the original image.



b) The chromosomes extracted at level 2 (16 by 16).

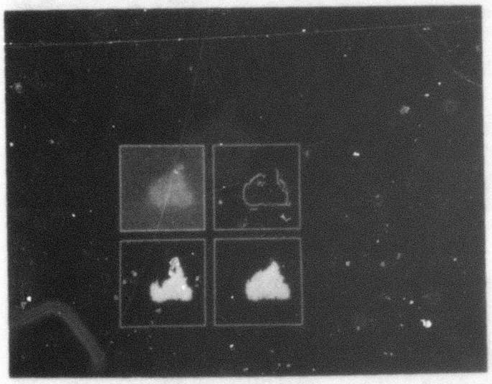


Figure 7. Top left: original FLIR image of a tank. Top right: edge image projected from the 8 by 8 level. Bottom left: the compact object found at the 8 by 8 level. Bottom right: the results of adding points to fit the edge data.

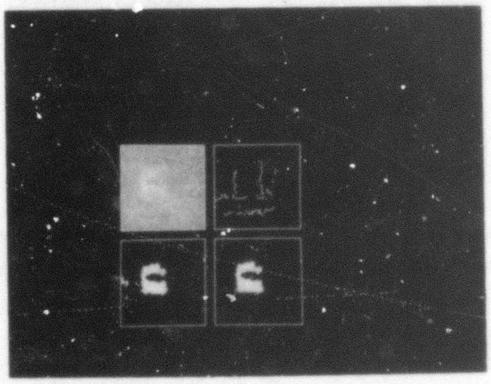


Figure 8. Top left: original FLIR image. Top right: edge image projected from the 8 by 8 level. Bottom left: compact object without pruning. Bottom right: compact object after pruning.

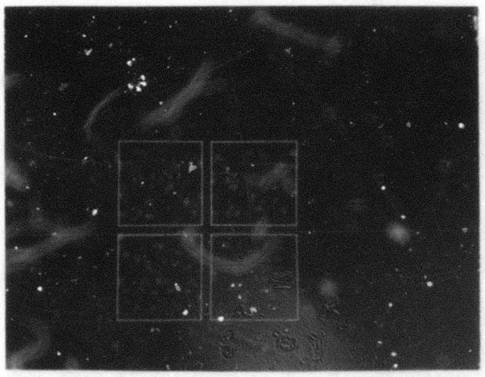
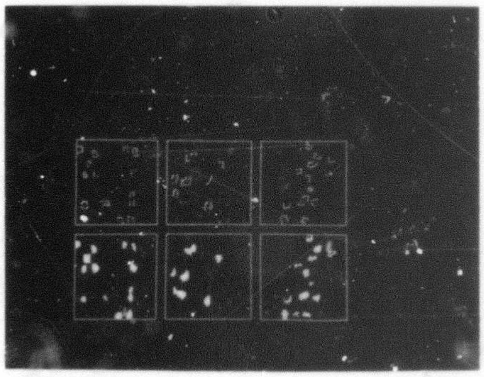
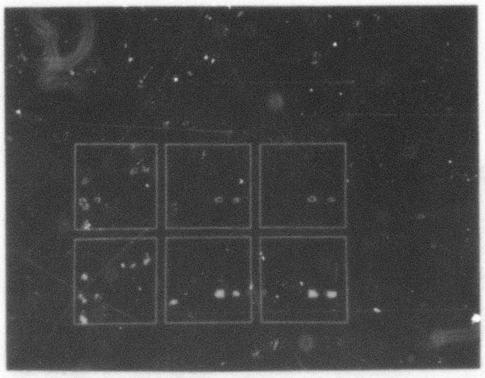
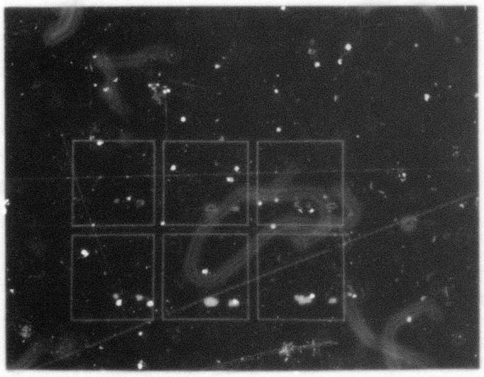


Figure 9. The results of running the process when the objects are found only at the base level of the pyramid (no thresholding).







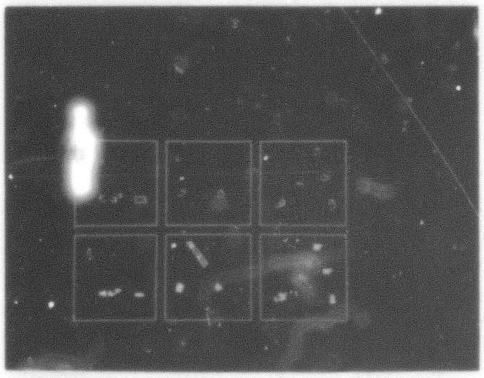
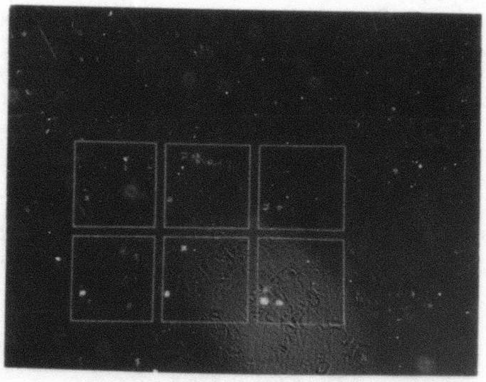
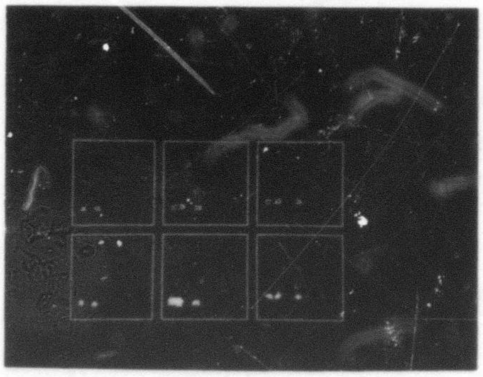


Figure 10. a-s. The results of running the process with the edge magnitude thresholded at 15.

a b c   d e f

g h i   j k ℓ





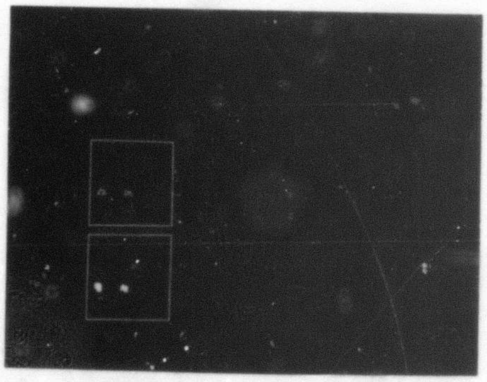


Figure 10, cont'd.:

m n o   p q r

s

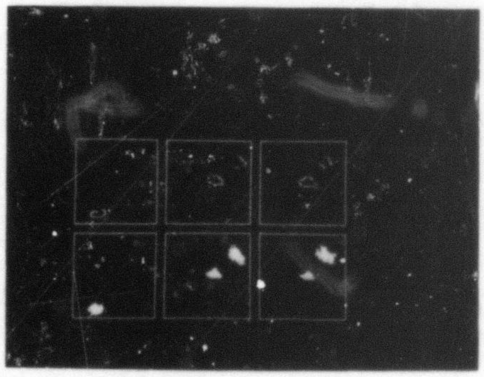
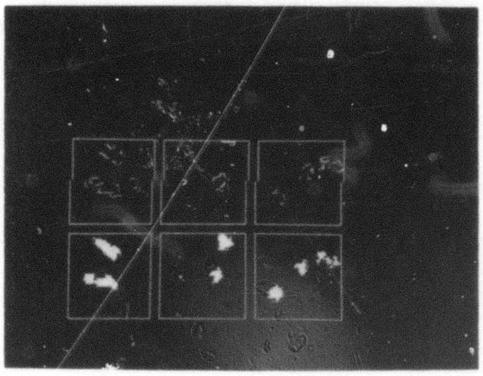


Figure 11. a-i. The results of running the process on images from Figure 9 when the objects are extracted at level 1 (32 by 32).

a b c



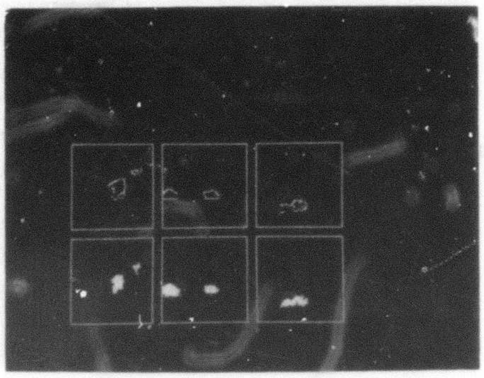
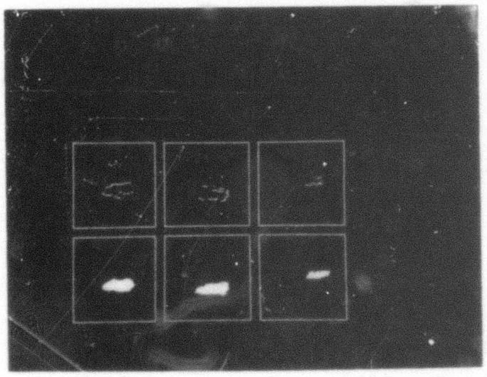
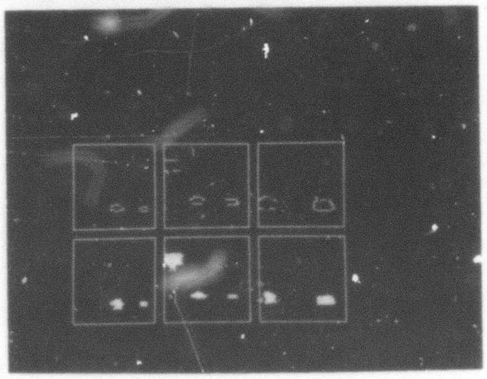


Figure 11, cont'd.: d e f   g h i





a b c   d e f

Figure 12. a-f. The results of running the process on images where the objects are extracted at level 2 (16 by 16).

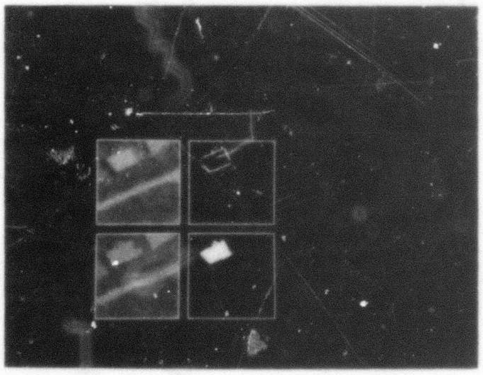


Figure 14. Part of a suburban scene with a road and a house. The house is compact enough to be extracted, but the road is not.

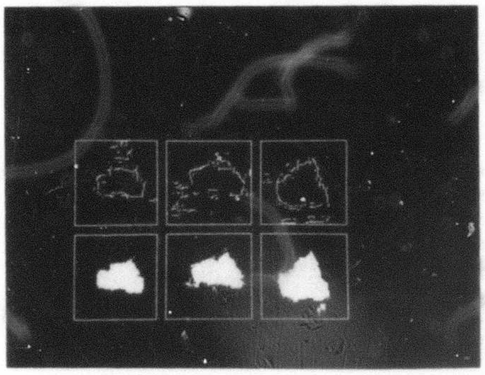
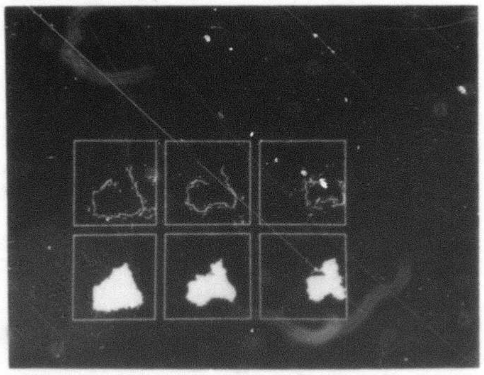
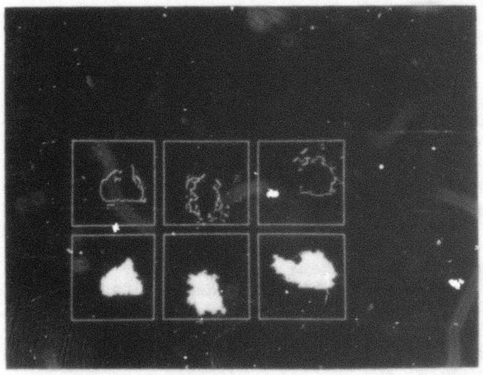
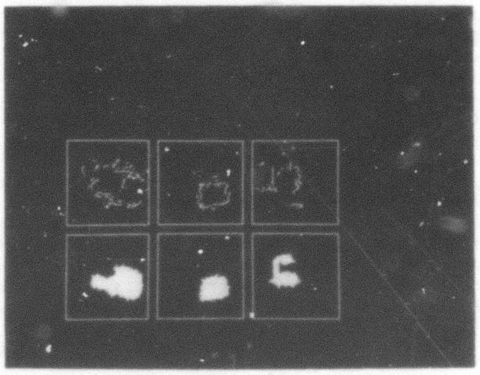
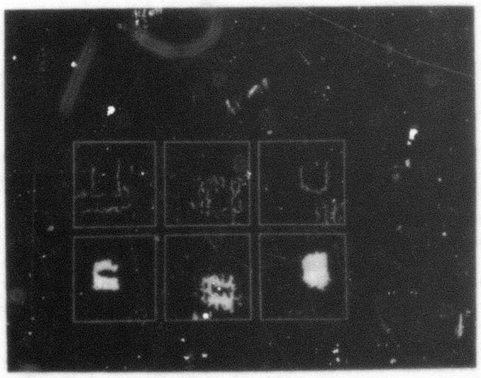
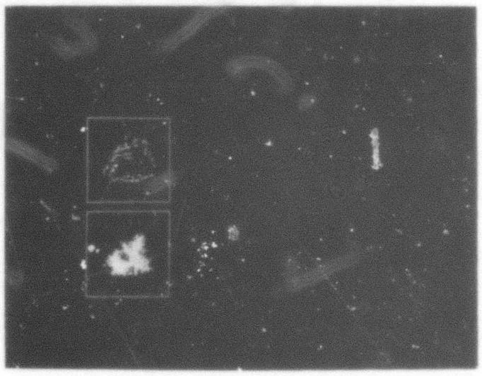







Figure 13. a-p. The results of running the process on images where the objects are extracted at level 3 (8 by 8).

a b c d e f

g h i j k ℓ

m n o p

AD-P000 113

UNFORCED IMAGE PARTITIONING BY WEIGHTED PYRAMID LINKING

Tsai-Hong Hong
Azriel Rosenfeld

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

This paper describes a method of image segmentation that creates a partition of the image into compact, homogeneous regions using a parallel, iterative approach that does not require immediate forced choices. The approach makes use of a "pyramid" of successively reduced-resolution versions of the image. It defines link strengths between pairs of pixels at successive levels of this pyramid, based on proximity and similarity, and iteratively recomputes the pixel values and adjusts the link strengths. After a few iterations, the link strengths stabilize, and the links that remain strong define a set of subtrees of the pyramid. Each such tree represents a compact (piece of a) homogeneous region in the image; the leaves of the subtree are the pixels in the region, and the size of the region depends on how high the root of the tree lies in the pyramid. Thus the trees define a partition of the image into (pieces of) homogeneous regions.

1. Introduction

Most of the existing methods of image segmentation [1,2] are based on forced-choice decisions. In methods that classify pixels into subpopulations, we must decide to which class each pixel belongs. In methods that partition the image into homogeneous regions using splitting and merging processes, we must decide, for each current region, whether to split it, or whether to merge it with a neighboring region (and if so, with which one). This forced-choice aspect of segmentation is undesirable, since many of the decisions may be wrong, particularly when they are made on the basis of very little information, and it is difficult to undo the effects of wrong decisions.

In segmentation by pixel classification, a "relaxation" approach [3] can be used to defer the classification decisions until more information is available. In this approach we compute a degree of membership for each pixel in each class, or a "probability" that it belongs to each class; and we then iteratively adjust these membership values, based on the values at neighboring pixels and the compatibilities of the various possible combinations of class memberships of pairs of neighbors. After a few iterations, the membership values stabilize, with some values becoming or remaining relatively high and others becoming very low, so that it becomes easy to make the final classification decisions.

Segmentation by partitioning into homogeneous regions - e.g., regions of approximately constant value - is generally more powerful than segmentation by pixel classification, because the information on which it is based is computed over regions rather than ("myopically") over small neighborhoods of pixels. Thus it would be desirable to develop a region-based segmentation scheme in which decisions are not made immediately. This paper defines such a scheme and gives examples of the results obtained when it is applied to various types of images. Section 2 describes the general principles of this scheme and compares it with some related approaches; Section 3 discusses the algorithm; and Section 4 presents experimental results.

2. Weighted pyramid linking

Our approach to unforced image partitioning makes use of a "pyramid" of successively reduced-resolution versions of the given image, say of sizes 2^n by 2^n, 2^{n-1} by 2^{n-1},..., 2x2. The base of the pyramid (level 0) is the input image, and each successive level is constructed by averaging 4 by 4 blocks of pixels on the level below, where the blocks overlap 50% in x and in y. (For convenience, each level is regarded as cyclically closed, so that its top row is adjacent to its bottom row and its left column to its right column.) Thus each pixel on a given level has 16 "sons" on the level below (if any) that contribute to its average, and 4 "fathers" on the level above (if any) to whose average it contributes. This type of pyramid has also been used for segmentation purposes by other investigators; e.g., see the work of Hanson and Riseman described in [4].

The basic idea in our approach is to define link strengths between "neighboring" pixels (i.e., father/son pairs) on adjacent levels of the pyramid, based on the similarity (in value) and proximity (in (x,y) coordinates) of each such pair. We then recompute the pixel values (at the levels above the base) as <u>weighted</u> averages of their sons'

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Clara Robertson in preparing this paper.

values, where the weights depend on the link strengths. These new values define new link strengths, and the process is iterated. (The details of the algorithm will be given in the next section.) After a few iterations, the link strengths stabilize, and the links that remain strong define subtrees of the pyramid. As it turns out, each such tree defines a compact homogeneous region in the image, where the leaves of the tree are the pixels belonging to the region, and the height of the tree corresponds to the region size (the larger the region, the higher the root of its tree lies in the pyramid). Thus the weighted links can be used to define a partition of the image into compact homogeneous regions. Note that this partition is not defined immediately, but only after the link weights have stabilized.

To see intuitively why this approach should work, consider the case of a homogeneous compact region on a homogeneous background. Pixels in the interior of the region (or background) will link strongly to all their fathers, since these fathers' values are averages of image blocks that lie in the same regionl A pixel near the region border, however, will link more strongly to a father that lies inside the region than to one that lies partly outside, since it is more similar in value to the former. Thus when we recompute the fathers' values, a father whose image block lies mostly inside the region will get closer in value to the average of the region, since it is more strongly linked to its sons that lie in the region than to those that lie in the background; and conversely. This makes its links to the former sons even stronger, and to the latter even weaker, so that the link strengths and values should converge. Now consider a pixel whose block lies mostly inside the region, but whose fathers' blocks all lie mostly outside, because they are bigger than the region. By the argument just given, the pixel's value should tend toward the region average, while its fathers' values should tend toward the background average, so that the pixel does not remain strongly linked to any of its fathers, and becomes the root of a tree representing (a compact portion of) the region.

It is of interest to compare this approach to some earlier segmentation schemes based on pyramid linking or on link strengths. In [5-6] link strengths are computed between each father/son pair, but we keep only the strongest of the four links between a pixel and its fathers. We then recompute the pixel values allowing only those sons that are linked to a pixel to contribute to its value; recompute the link strengths based on these new values; and iterate the process. Note that in this scheme every pixel must link to one of its fathers; thus the links define precisely four trees, rooted at the top (2x2) level, so that the image is segmented into precisely four sets of pixels. These sets do not correspond to compact regions, but do tend to correspond to homogeneous subpopulations of pixels. Thus the segmentation scheme of [1-2] is more like a pixel clustering and classification scheme than an image partitioning scheme; and it also makes forced choices immediately, since it keeps only the strongest upward link from each pixel. Extensions of this scheme to segmentation based on color or texture, and to waveform or contour segmemtation, are described in [7-10].

A pyramid linking method which does make use of all the link strengths, rather than discarding all but the strongest upward link, is described in [11]. However, in this method the link strengths are normalized so that they sum to 1; thus here too the links are forced to extend upward from every pixel (divided among its fathers appropriately) all the way to the top level. In fact, the link strengths tend to converge to 0 or 1 where the process is iterated, so that this method too defines a segmentation of the image into four subpopulations of pixels, rather than a partition into regions.

A weighted pixel linking scheme not involving a pyramid is described in [12]. Here a link strength is computed for each pair of neighboring pixels based on their closeness in value. The image is then smoothed by replacing each pixel with the average of its neighbors, weighted by their link strengths. Using these new values, the link strengths are recomputed, and the process is iterated. This tends to produce a very high-quality smoothing, and the links that remain strong could be used to define a segmentation of the image into homogeneous regions; but this method would not always be reliable, since it is based on small neighborhoods. The method defined in this paper is analogous to the scheme in [12], but using "vertical" links (between fathers and sons) in a pyramid, rather than "horizontal" links (between brothers) in an image at a single resolution. Our method could be generalized to make use of horizontal as well as vertical link strengths, but we shall not pursue this possibility here.

3. The algorithm

The algorithm is initialized, as mentioned earlier, by building the pyramid using unweighted averaging of 4x4 blocks that overlap 50% horizontally and vertically. Alternatively, we could use nonoverlapping 2x2 blocks (for the initialization only; a pixel still has 16 sons in the subsequent steps), or we could use the median instead of the mean; but these variations were found to make little difference in the results.

Let $v(P)$ denote the value of pixel P in the pyramid, say on level ℓ. Initially, if $\ell=0$ this is the gray level of an input pixel, and if $\ell>0$ it is the mean of the values of P's 16 sons. Let $\sigma(P)$ be the standard deviation of these sons' values (or if $\ell=0$, we take σ to be a constant; we used 5 in our experiments).

Let P^* be one of the fathers of P. The link strength between P and P^* is defined by

$$w(P,P^*) \equiv (1+d(P,P^*))\frac{\exp(-\frac{1}{2}[\frac{v(P)-v(P^*)}{\sigma(P)}]^2)}{\sqrt{2\pi}\ \sigma(P)}$$

In this expression, the first factor depends on the distance between (the centers of) P and P*; d is taken to be 3 for the closest father, 1 for the farthest, and $\sqrt{5}$ for the other two. (It can be verified that these are proportional to the Euclidean distances between the centers.) This factor makes the sets of pixels that belong to a given tree more compact; if it is omitted, these sets become more irregular in shape. The factors $\frac{1}{\sigma(P)}$ reflect the (non) variability of the sons of P; if they are highly variable, P does not link strongly to any of its fathers. Finally, the exp factor depends on the similarly in value of P and P*; if they are very dissimilar, the link is weak.

We now want to recompute the pixel values at levels >0 as weighted averages of their sons' values, where the weights depend on the link strengths. Note first that the weight given to a son must also depend on the (weighted) "area" of the image represented by that son; for example, if one son had unit strength links (down through successive levels) to a single image pixel, and zero strengths to all its other descendants, we would not want to give it as much weight as a son that had high-strength links to many image pixels. Let a(P) be the "area" of pixel P; initially, for a pixel at level ℓ, we have $a(P) = 2^{2\ell}$, since P represents a 2^{ℓ} by 2^{ℓ} image block. Subsequently, let a(P') be the area of a son P' of P, and let w(P',P) be the link strength between them. Then $a(P) = \sum_{P'} w(P',P)a(P')/W(P')$ (where the sum is over the sons of P' of P); here $W = \sum_{P'*} w(P',P*)$ (the sum being over the fathers P'* of P'). Note that in computing a(P) we are actually using normalized weights, i.e., $\sum w(P',P'*)/W=1$. This is because it seems reasonable that the "area" of a pixel should be distributed among its fathers in a normalized fashion, in order to insure that the total "area" of all pixels at a given level remains equal to the area of the image.

Finally, the new value of pixel P is given in terms of its sons' values by

$$v(P) = \frac{\sum_{P*} v(P')a(P')w(P',P)}{\sum_{P'} a(P')w(P',P)}$$

where the sums are over the sons P' of P. Similarly, the new standard deviation is given by

$$\sigma(P) = \sqrt{\frac{\sum_{P'} (v(P) - v(P'))^2 a(P')w(P',P)}{\sum_{P'} a(P')w(P',P)}}$$

The process is iterated; in our experiments, only two or three iterations were necessary.

After the desired number of iterations, we call a pixel a "root" if it is on the top level (2x2), or if the sum of its link strengths to all its fathers is negligible (in our experiments: $\leq 10^{-5}$). The nonroot pixels are then assigned to trees by using only their most strongly linked fathers.

4. Experiments

The algorithm just described was applied to the three images shown in Figure 1: photomicrographs of some chromosomes (right) and blood cells (left), and an infrared image of a tank. Each image is 64x64 pixels; thus the top (2x2) level of the pyramid is level 5. At each iteration, the gray level displayed for each pixel is the value at the root of its tree. We see that even after a single iteration, the trees define a decomposition of the image into regions having a small set of values; and in one or two more iterations the set of values is reduced even further.

Table 1 lists the root nodes at each level, and their values, for each image for as many iterations as were needed until there was no further change in the set of roots. We see that the more complex the image, the more iterations are required for the set of roots to stabilize; but that even for the most complex image, the changes after the first two or three iterations have little effect on the segmentation of the image.

Figure 2 shows printouts of the displayed images after the first (parts a-c) and last (parts d-f) iterations, where the value printed in each region identifies the root of the tree to which it belongs; the digit is the level, and the letters are used to distinguish the roots at that level. We see that after a few iterations, the leaves of each tree define a small set of compact regions. As Table 1 indicates, regions that are compact pieces of a single homogeneous region have nearly the same value. Note that because of the coordinate wraparound, regions on opposite sides of the image may belong to the same tree.

5. Concluding remarks

We have exhibited a method of segmenting an image into compact honogeneous regions by constructing links between "neighboring" pixels at consecutive levels of a "pyramid".

An important feature of this method is that each region is represented by a tree having the pixels of the region as leaves. The height of this tree is proportional to the log of the region size. Thus, even for large regions, all the pixels in the region are relatively closely linked to the root of the tree, and hence to each other. The pyramid structure makes it possible for information to propagate between different parts of a region relatively rapidly. Moreover, the root of the tree can be used as a node to represent the region in various region-level relational structures. Thus the tree constitutes a transition between the pixel-level representation of the region and more abstract representations.

Another important feature of our method is that the trees are produced by a cooperative process in which link strengths are iteratively adjusted. Under this process, root pixels representing regions become easy to recognize, because their link strengths to their fathers all become negligible. They are harder to recognize in the original pyramid, where the pixels (especially at higher levels) represent mixtures of image pixels, so that the link strengths are not initially negligible.

Image processing and segmentation techniques based on "local" operations performed in a pyramid can be implemented very rapidly in parallel on a tree-structured cellular processor [13]. It is possible that processes of this type also play a role in biological visual systems, where the input image is represented at a range of resolutions [14].

REFERENCES

1. A Rosenfeld and A. C. Kak, Digital Picture Processing, second edition, Academic Press, New York, 1982, Chapter 10.

2. T. Pavlidis, Structural Pattern Recognition, Springer, New York, 1977.

3. A. Rosenfeld and L. S. Davis, Cooperating processes for low-level vision: a survey, Artificial Intelligence 17, 1981, 245-263.

4. A. Klinger and S. L. Tanimoto, Structured Computer Vision, Academic Press, New York, 1980.

5. P. J. Burt, T. T. Hong and A. Rosenfeld, Image segmentation and region property computation by cooperative hierarchical computation, IEEE Trans. Systems, Man, Cybernetics 11, 1981, 802-809.

6. T. Silberberg, S. Peleg, and A. Rosenfeld, Multi-resolution pixel linking for image smoothing and segmentation, TR-977, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, November 1980.

7. T. H. Hong and A. Rosenfeld, Multiband pyramid linking, TR-1025, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, March 1981.

8. M. Pietikäinen and A. Rosenfeld, Image segmentation by texture using pyramid node linking, IEEE Trans. Systems, Man, Cybernetics, 11, 1981, 822-825.

9. M. Pietikäinen, A. Rosenfeld, and I. Walter, Split-and-link algorithms for image segmentation, Pattern Recognition 14, 1982, in press.

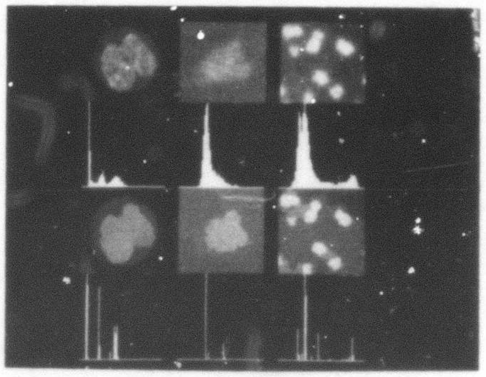
10. K. A. Narayanan and A. Rosenfeld, Approximation of waveforms and contours by one-dimensional pyramid linking, Pattern Recognition 14, 1982, in press.

11. K. A. Narayanan, S. Peleg, A. Rosenfeld, and T. Silberberg, Iterative image smoothing and segmentation by weighted pyramid linking, TR-989, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1980.

12. J. O. Eklundh and A. Rosenfeld, Image smoothing based on neighbor linking, IEEE Trans. Pattern Analysis Machine Intelligence 3, 1981, 670-683.

13. C. R. Dyer, A VLSI pyramid machine for hierarchical parallel image processing, Proc. PRIP '81, August 1981, 381-386.

14. L. Uhr, Psychological motivation and underlying concepts, in [4], 1-30.



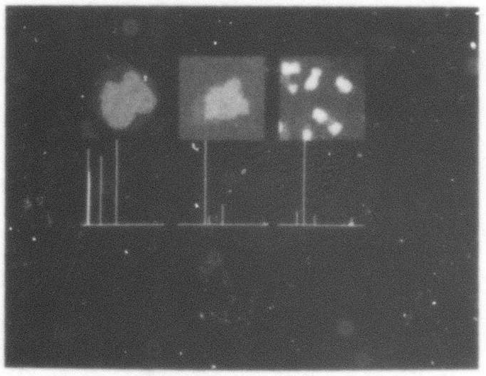
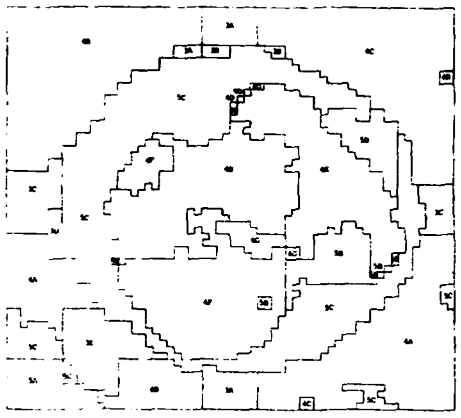


Figure 1. a) Input images and their histograms
b) Results after one iteration
c) Results after last iteration



Fir re 2a

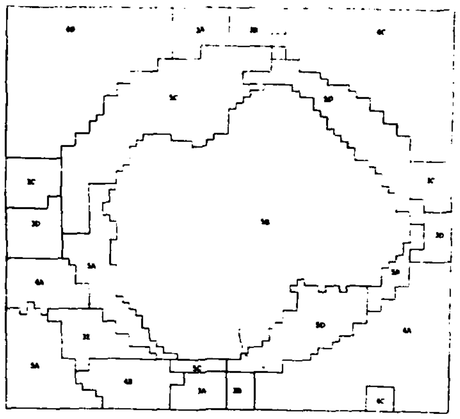


Figure 2d

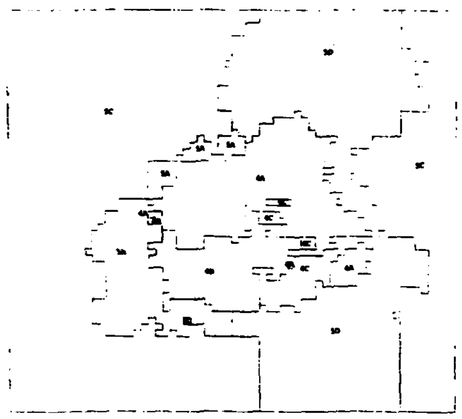


Figure 2b

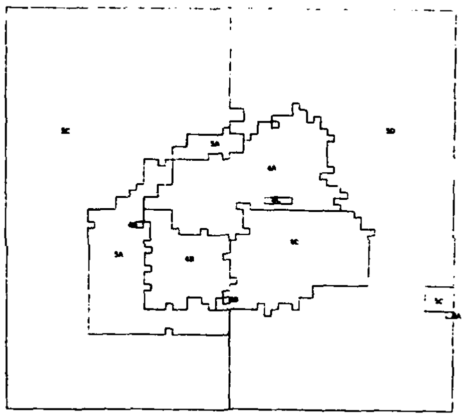


Figure 2e

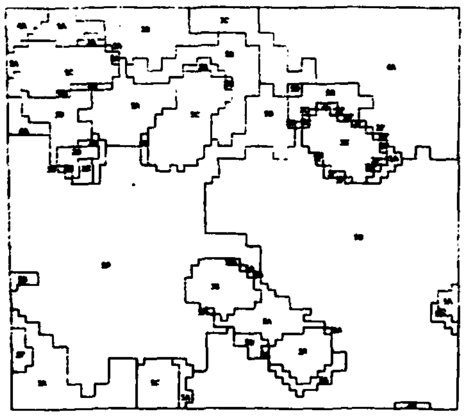


Figure 2c

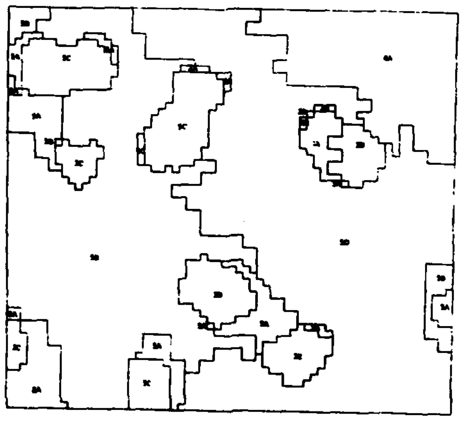


Figure 2f

Table 1. Root nodes and their values at each iteration for the three images. The root labels (see Figure 2) are given only for the first and last iterations.

(a) Cell image; there were no changes in the set of root nodes after the third iteration.

Iteration	Level	Root Label	Coordinates	Value
1	3	A	(0,3)	5.65
		B	(0,4)	6.87
		C	(3,0)	5.68
		D	(4,0)	6.08
		E	(6,1)	5.85
	4	A	(3,3)	5.56
		B	(0,0)	5.22
		C	(0,3)	5.34
		D	(1,1)	26.89
		E	(1,2)	28.27
		F	(2,1)	29.14
		G	(2,2)	29.64
	5	A	(1,0)	15.63
		B	(1,1)	23.96
		C	(0,0)	15.28
		D	(0,1)	15.61
2	3		(0,3)	5.34
			(0,4)	5.67
			(3,0)	5.63
			(4,0)	5.84
			(6,1)	5.80
	4		(3,3)	5.52
			(0,0)	5.23
			(0,3)	5.35
			(2,2)	28.70
	5		(1,0)	15.27
			(1,1)	25.14
			(0,0)	15.22
			(0,1)	15.27
3	3		(0,3)	5.32
			(0,4)	5.49
			(3,0)	5.63
			(4,0)	5.80
			(6,1)	5.78
	4		(3,3)	5.53
			(0,0)	5.23
			(0,3)	5.35
	5		(1,0)	15.18
			(1,1)	26.39
			(0,0)	15.17
			(0,1)	15.17
4	3	A	(0,3)	5.31
		B	(0,4)	5.48
		C	(3,0)	5.63
		D	(4,0)	5.79
		E	(6,1)	5.78
	4	A	(3,3)	5.53
		B	(0,0)	5.23
		C	(0,3)	5.36
	5	A	(1,0)	15.17
		B	(1,1)	26.78
		C	(0,0)	15.16
		D	(0,1)	15.16

(b) Tank image; no changes after the second iteration. Note that one of the roots is a single pixel.

Iteration	Level	Root Label	Coordinates	Value
1	4	A	(1,2)	33.59
		B	(2,1)	34.40
		C	(2,2)	34.59
	5	A	(1,0)	23.49
		B	(1,1)	23.40
		C	(0,0)	20.92
		D	(0,1)	20.96
2	0		(48,63)	43.00
	4		(1,2)	33.45
			(2,1)	33.81
			(2,2)	33.64
	5		(1,0)	23.18
			(1,1)	22.93
			(0,0)	20.37
			(0,1)	20.41
3	0	A	(48,63)	43.00
	4	A	(1,2)	32.96
		B	(2,1)	33.34
		C	(2,2)	33.03
	5	A	(1,0)	22.90
		B	(1,1)	22.33
		C	(0,0)	20.25
		D	(0,1)	20.29

(c) Chromosome image; no changes after the eighth iteration.

Iteration	Level	Root Label	Coordinates	Value
1	2	A	(15,14)	32.82
		B	(2,7)	37.80
		C	(4,10)	37.51
		D	(5,2)	48.73
		E	(6,2)	49.51
		F	(13,0)	53.54
	3	A	(0,1)	35.84
		B	(0,2)	17.16
		C	(0,4)	14.93
		D	(2,0)	23.94
		E	(2,5)	55.38
		F	(2,6)	47.02
		G	(5,3)	53.99
		H	(6,5)	51.50
	4	A	(0,3)	14.03
		B	(1,0)	36.74
	5	A	(1,0)	28.92
		B	(1,1)	19.19
		C	(0,0)	53.83
		D	(0,1)	19.68

(Cont'd. on next page)

2	?		(15,14)	31.31
			(2,7)	37.86
			(4,10)	36.87
			(5,2)	48.48
			(6,2)	48.82
			(6,6)	32.01
			(13,0)	52.93
	3		(0,1)	35.28
			(0,2)	17.32
			(2,5)	55.32
			(2,6)	47.45
			(5,3)	53.77
			(6,5)	52.25
			(7,1)	21.71
	4		(0,3)	14.05
			(1,0)	37.11
	5		(1,0)	28.29
			(1,1)	19.17
			(0,0)	54.19
			(0,1)	19.38
3	2		(15,14)	29.23
			(2,7)	38.03
			(4,10)	36.65
			(5,2)	48.34
			(6,2)	48.53
			(6,6)	30.35
			(13,0)	52.67
	3		(0,1)	34.21
			(0,2)	17.46
			(2,5)	55.23
			(2,6)	49.17
			(5,3)	53.62
			(6,5)	52.21
			(7,1)	21.39
	4		(0,3)	14.06
	5		(1,0)	27.65
			(1,1)	19.26
			(0,0)	54.15
			(0,1)	19.23
4	2		(15,14)	26.71
			(2,7)	38.13
			(4,10)	36.50
			(13,0)	52.57
	3		(0,1)	32.97
			(0,2)	17.59
			(2,5)	55.16
			(2,6)	49.19
			(5,3)	53.51
			(5,4)	33.21
			(6,5)	52.12
			(7,1)	21.27
	4		(0,3)	14.07
	5		(1,0)	27.28
			(1,1)	19.36
			(0,0)	54.09
			(0,1)	19.12
5	2		(2,7)	38.12
			(4,10)	36.22
			(13,0)	52.52
	3		(0,2)	17.70
			(2,5)	55.10
			(2,6)	50.48
			(5,3)	53.44
			(5,4)	32.17
			(6,5)	52.03
	4		(0,3)	14.08
	5		(1,0)	27.04
			(1,1)	19.45
			(0,0)	54.04
			(0,1)	19.06
6	2		(2,7)	38.03
			(4,10)	35.88
			(13,0)	52.50
	3		(2,1)	28.54
			(2,5)	55.04
			(2,6)	52.32
			(5,3)	53.38
			(6,5)	51.97
	4		(0,3)	14.08
			(1,0)	36.96
	5		(1,0)	26.81
			(1,1)	19.49
			(0,0)	54.01
			(0,1)	19.04
7	2		(2,7)	37.87
			(4,10)	35.53
			(13,0)	52.49
	3		(2,5)	54.97
			(2,6)	54.00
			(5,3)	53.34
			(6,5)	51.92
	4		(0,3)	14.09
			(1,0)	36.95
	5		(1,0)	26.92
			(1,1)	19.51
			(0,0)	53.99
			(0,1)	19.05
8	2		(2,7)	37.65
			(4,10)	35.16
			(13,0)	52.49
	3		(2,5)	54.93
			(2,6)	54.69
			(3,1)	48.04
			(5,3)	53.31
			(6,5)	51.89
	4		(0,3)	14.10
			(1,0)	36.96
	5		(1,0)	26.95
			(1,1)	19.51
			(0,0)	53.99
			(0,1)	19.05
9	2	A	(2,7)	37.36
		B	(4,10)	34.75
		C	(13,0)	52.50
	3	A	(2,5)	54.90
		B	(2,6)	54.94
		C	(3,1)	48.07
		D	(5,3)	53.29
		E	(6,5)	51.86
	4	A	(0,3)	14.10
		B	(1,0)	36.96
	5	A	(1,0)	26.84
		B	(1,1)	19.49
		C	(0,0)	53.99
		D	(0,1)	19.05

AD-P000 114

CONSTRAINT INTERACTION IN SHAPE-FROM-SHADING ALGORITHMS

C.M. Brown, D.H. Ballard, and O.A. Kimball

Computer Science Department
University of Rochester
Rochester, NY 14627

ABSTRACT

After a review of shape-from-shading basics, we describe experiments in constraint interaction. Boundary conditions can interact significantly to influence computed shape. Interleaved, cooperating shape-from-shading and illuminant-direction calculations show constraint interaction in underdetermined computation of physical parameter images.

1. Background and Overview

The goal of a shape from shading algorithm is to derive 3-D shape information from a 2-D image in which the value at a point (the image irradiance) is proportional to the intensity of light reflected by a small foreshortened area of opaque surface (the scene radiance). Shape is a physical property of an object. In this work, shape refers to the mathematical specification of the surface -- its height as a function of two-dimensional position, say. We define the term parameter image to mean an image-like array of physical parameters, perhaps expressed in viewer centered coordinates. We use intrinsic image [Barrow and Tenenbaum 1978] to mean a parameter image expressed in object-centered coordinates. Shape, by almost any qualitative or quantitative definition, is determined by the large-scale variation of surface orientation over the object. This macroscopic variation is linked with the infinitesimal behavior of the surface -- its local orientation. Knowledge of local orientation is enough to allow the surface function to be reconstructed (up to a constant depth offset). Such an array of orientation is a shape parameter image; an array of heights is another.

One of our goals is to derive an orientation image from an input intensity image. The computation of an array of vector orientations from an array of scalar intensities is in general underdetermined. Usually the computation is made possible through assumptions about the continuity of physical objects (surface smoothness) and about the imaging situation (the composition and placement of the illuminant and the reflectivity of the objects in the scene), taken with boundary conditions (a set of points at which orientation is known). Under these conditions the variation of intensity (the shading) in the image can indeed yield shape. This is one of the most influential ideas in modern computer vision [Horn 1970, 1975, 1977; Woodham 1978; Strat 1979; Brooks 1979; Barrow and Tenenbaum 1981; Ikeuchi and Horn 1981]. Good discussions of various approaches appear in [Strat 1979; Ikeuchi and Horn 1981; Grimson 1981]. Modern approaches (e.g. [Ikeuchi and Horn 1981]) use parallel-iterative techniques to minimize error terms arising from the smoothness and reflectance constraints. Grimson [1981] has studied algorithms for surface interpolation, deriving conditions for surface consistency functionals and investigating convergence properties. Bruss [1981] has studied the mathematical properties of the irradiance equation, which lies at the base of the shape-from-shading enterprise.

The human visual system seems capable of extracting shape with less explicit information than shape from shading mathematics needs. In particular, it may be unnecessary to know the imaging geometry (the illuminant direction) apriori. One factor here may well be the importance of local, differential measures (as opposed to global consistency) [Pentland 1982]. Another factor may be that parameter image calculations are mutually constraining, and that they can converge to consistent results despite seeming indeterminacy.

Our concern is with the interaction of parameter image computations, which is a large and interesting area. Depth information is known to influence lightness computations, for example [Gilchrist 1979]. Rigid body motion and shape can be computed from optical flow [Ballard and Kimball 1982]. One experiment we describe has to do with concurrent computations of illuminant direction and shape. Our input is an intensity image and a reflectance function for the surface, but no information about imaging geometry (such as a reflectance map or the illuminant direction). Our strategy is to interleave a standard relaxation computation for shape with a step that adjusts the light source position to be consistent with the developing shape results and the intensity data. The partial results should constrain each other, and the research is to determine how.

Unfortunately, the theory of first order partial differential equations guarantees (loosely) that given an intensity image, then for any reflectance map there is a solution (a 2-D height function of x,y) passing through any well-behaved strip (a curve whose height is a function of a curve in the x,y plane) that does not happen to be a characteristic strip (located along gradient maxima) [Bruss 1981]. Just about any intensity image is consistent with just about any illuminant direction.

Clearly then what makes the co-operating derivation of shape and illuminant direction interesting is the interaction, through an algorithm, of constraints on shape, on reflectance, and on orientations and intensities known apriori. We have constructed a flexible computational laboratory to explore algorithms and the interactions of such constraints in them. We give a brief review of orientation space parameterizations in Section 2, reflectance functions in Section 3, surface smoothness measures in Section 4, relevant shape-deriving techniques in Section 5, and parameter transforms in Section 6. Section 7 describes the implementation of the the algorithm for shape and illuminant direction determination. Section 8 gives experimental results.

2. Orientation Spaces

The representation and manipulation of directions is basic to the enterprise of deriving shape from shading. The orientation of a surface at a point is determined by the direction of its surface normal (a vector perpendicular to the surface) there. The Sections 2.1 - 2.4 briefly present useful orientation representations. Ikeuchi and Horn [1981] give a good brief treatment of spherical projections.

2.1 Polar Space (Gaussian Sphere)

The direction of a vector specifies a direction in space. If the vector is based at the origin, its components (the polar space coordinates of a point) are the direction numbers of the line from the origin through the point. If the vector is of unit length, then its Cartesian components (x,y,z) are direction cosines, and indicate a point on the unit (Gaussian) sphere, which is often taken to represent 3-D directions [Gauss 1965]. The points (0,0,1) and (0,0,-1) are respectively the z and -z poles of the Gaussian sphere.

This redundant three-dimensional (x,y,z) representation of directions has the advantage of intuitive clarity, good interpolation properties [Barrow and Tenenbaum 1981], and a straightforward geometrical interpretation [Brown 1979].

2.2 Gradient Space: Slant and Tilt

If a surface is represented as a function f(x,y) of two dimensions, then its gradients, or partial derivatives (p,q) at a point represent the orientation of the surface at that point.

$$p = \partial f / \partial x$$

$$q = \partial f / \partial y$$

Gradient space is the projection of half the Gaussian sphere onto a plane. The plane of gradient space is tangent to the Gaussian sphere at the z pole, and the point of projection is the origin (center) of the sphere. The equator is projected out to points at infinity in (p,q) space, and one hemisphere is not projected at all.

The slant of a surface is the amount of its inclination away from the z axis (along which a viewer is assumed to look). Slant is taken to be zero when the surface is perpendicular to the z axis (its normal is parallel to the z-axis), and infinity when it is parallel to the z-axis (its normal is perpendicular the z-axis). The slant of an orientation in (p,q) gradient space is its distance from the origin; all orientations on a circle centered at the origin have the same slant. Thus points around the equator of the Gaussian sphere represent orientations with infinite slant.

The tilt of a surface is the direction of its slant. In gradient space, the tilt of a surface with gradients (p,q) is measured by the angle between a fixed direction (perhaps the p axis) and the line joining the origin and (p,q). Tilt and slant are simply polar coordinates of gradient space.

Gradient space has been a popular tool in computer vision for some time [Horn 1970, 1975, 1977; Mackworth 1973; Huffman 1978]. Its advantages come from its dual relation to image space and its mathematical relation to the surface height function. (A 2-D array of (p,q) vectors may be integrated directly to recover f(x,y) to within a constant depth displacement.) However, it only represents half the space of orientations, it has a singularity at an important angle (that normal to the z-axis viewing direction), and of course it does not lend itself to linear interpolation for directions. To convert from gradient space to polar space, use

$$(x,y,z) = (-p, -q, 1) \,/\, \sqrt{(p^2 + q^2 + 1)}.$$

2.3 Stereographic Space

If the Gaussian sphere is projected, again onto a plane tangent to its z-pole, with the -z pole as the point of projection, the plane has orientations represented in (f,g) stereographic space. Points on the equator (surfaces of infinite slant) project to a circle of radius two, and the whole Gaussian sphere is represented on the infinite (f,g) plane. The main advantage of stereographic space is that it represents more orientations than does gradient space and only has a singularity at a single angle. It yields complicated analytic expressions, and again linear interpolation is not exact [Ikeuchi and Horn 1981]. To convert from gradient space to stereographic space, use

$$f = 2p[\sqrt{(p^2 + q^2 + 1)}] \,/\, (p^2 + q^2)$$

$$g = 2q[\sqrt{(p^2 + q^2 + 1)}] \,/\, (p^2 + q^2).$$

2.4 Spherical Space (Longitude and Co-latitude)

This orientation space is familiar from geography and spherical coordinates. A point on the Gaussian sphere is identified by (θ, φ), its longitude and co-latitude ($\pi/2$ - latitude). The equator of the Gaussian sphere is at $\theta = \pi/2$. To convert from gradient space to spherical space, use

$$\theta = \arctan\left(\sqrt{(p^2 + q^2)}\right)$$

$$\varphi = \arctan(p/q).$$

3. Reflectance

Generally, the apparent brightness of a surface under illumination depends on the surface and on the geometry of the viewing situation. This dependence is captured in the Bi-directional Reflectance Function (BDRF) [Nicodemus et al. 1977]. The BDRF for any surface is a function of the incidence, emittance (viewing), and phase angles of light on a surface patch, which themselves are defined relative to the surface normal. Horn and Sjoberg [1978] explore the BDRF and derive simplified forms of it from physical first principles.

When the imaging geometry is fixed (orthographic projection, fixed viewer, fixed object of fixed BDRF, and fixed light sources), the reflectance properties of a surface become a function just of its orientation with respect to the viewer. This dependence is made explicit in a reflectance map

R: Orientation --> Intensity,

which assigns an image brightness to each surface orientation.

The irradiance equation expresses the relation between image intensity and surface orientation:

$$I(i,j) = R(\text{Orientation}(i,j)). \qquad (3.1)$$

The reflectance map is a powerful tool. Looked at as a contour map of brightnesses, it restricts the possible orientations at point (i,j) to the (usually one dimensional) iso-brightness contour of value I(i,j). The simplest and most common reflectance function is the Lambertian, or matte-finish. A planar Lambertian surface looks equally bright from any direction (any angle of emittance), depending only on the angle of incidence of light. This is because the amount of light emitted by a small patch of surface increases as the cosine of the angle of emittance, while the amount of surface subtended by a solid angle increases as the reciprocal of the cosine of the angle of emittance. Thus the contribution of the emittance angle is cancelled, and the perceived brightness of the surface depends only on the angle of incidence. The computational simplicity of the Lambertian reflectance function make it a favorite of computer graphics and computer vision, but the understanding of reflectance functions is becoming quite sophisticated [Cook and Torrance 1981; Whitted 1980].

4. Smoothness

One of the most useful constraints on object shape (unless their geometric form is known [Woodham 1978a]) is the assumption that object surfaces are "smooth". Discontinuous variation in surface normals then only occurs between objects, not within objects, and thus is a relatively rare event. Ultimately, the smoothness constraint is incorporated into an error term that penalizes non-smooth surfaces and encourages smooth ones. Thus the choice of technical definition for smoothness affects the performance of shape-from-shading algorithms (Section 5.3). Barrow and Tenenbaum [1981] present useful smoothness measures for both surfaces and space curves. Ikeuchi and Horn [1981] present a good discussion of these issues.

Brooks [1979] and Strat [1979] independently developed algorithms based on a smoothness criterion involving the existence and continuity of second partial derivatives of surface height z. This criterion seems overly stringent, and its application is dependent on the gradient space representation of orientations.

Ikeuchi and Horn [1979] use a criterion whose application is representation-independent. A smooth surface is a C_0 and C_1 function: its depth has no discontinuities, nor do its first partial derivatives. Because the map from image coordinates to surface orientation is continuous, this criterion can be used for any continuous orientation space.

With this criterion a natural smoothness error term is the sum of four squared differences of components of neighboring points in orientation space.

$$s_{ij} = [(f_{i+1,j} - f_{ij})^2 + (f_{i,j+1} - f_{ij})^2 + (g_{i+1,j} - g_{ij})^2 + (g_{i,j+1} - g_{ij})^2]/4. \qquad (4.1)$$

The analytical minimization of the s_{ij} error admits of solution by a relaxation algorithm (Section 5.4). The resulting linear interpolation *of orientations is appealing and* mathematically justified, and we use it in the work reported here. However, the very simple form of the iterative equations (eq. (5.3.4)) invites the substitution of nonlinear techniques to incorporate other versions of the smoothness constraint.

5. Shape from Image

This section briefly outlines a few "shape from..." techniques that are relevant to our work. Good discussions of these techniques appear in [Strat 1979; Ikeuchi and Horn 1981].

5.1 Multiple Images -- Photometric Stereo

This technique [Woodham 1978b, Ikeuchi to appear] is very simple in concept. For a given imaging geometry and image intensity at a point (i,j), the irradiance equation (3.1) constrains the orientations to the (graph of the) I(i,j) iso-brightness contour in the reflectance map. Another imaging geometry similarly yields another locus in another reflectance map. The orientation of the surface at (i,j) must be consistent with both intensities; it lies in the intersection of both loci. The easiest way to change the imaging geometry is to move the light source -- recall that changing the viewpoint will not alter the brightness of Lambertian surfaces. Three equations are needed to determine the reflectance and the unit normal, so we need three light source positions at which point (i,j) is not in shadow. Let each source position vector be denoted by n_k, $k = 1,...,3$, and rewrite equation (3.1) as

$$I_k(i,j) = r(n_k \cdot n_s),\ k = 1,2,3, \qquad (5.1.1)$$

or in matrix form as

$$I = rNn_s \qquad (5.1.2)$$

where

$$I = [I_1(i,j),\ I_2(i,j),\ I_3(i,j)]^T \qquad (5.1.3)$$

and

$$N = \begin{matrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{matrix} \qquad (5.1.4)$$

If the three illuminant directions n_i are not coplanar, N is nonsingular and we can solve for r and n_s, for example by

$$r = |N^{-1}|I, \qquad (5.1.5)$$

$$n_s = (1/r)\ N^{-1}I. \qquad (5.1.6)$$

Figure 1 shows three source positions with a Lambertian surface and gradient space orientation representation, and how the iso-brightness contours of the three corresponding reflectance maps intersect at a unique (p,q).

5.2 Local Analysis -- Parameters from Differential Shading

Local shading is an important source of shape parameters and illuminant direction information. Local clues provide constraints that can be used with considerable effect by global relaxation algorithms.

The thesis of Pentland [1982a, 1982b] thoroughly explores this use of local (differential) shading measures. Some (non-statistical) conclusions from local measures are immediate, since no global calculations like constraint propagation or relaxation are involved. Other (statistical) conclusions can be reached by accumulating local evidence, still not using iterative relaxation. They yield maximum-likelihood estimates of imaging parameters.

Pentland shows how to determine a maximum-likelihood estimator for illuminant direction under certain reasonable assumptions about the imaged domain, and compares the results of his algorithm to human performance. Given the illuminant direction, the differential shading measures can provide approximations to surface curvature properties. The local determination of surface tilt (direction of slant) means that normals are approximately constrained to have only one degree of freedom -- this can be quite important in a global relaxation algorithm. The slant determination and the illuminant direction calculation are statistical. These estimates are obvious choices to initialize parameters for global relaxation computations.

5.3 Global Relaxation -- Shape from Line Drawings

Barrow and Tenenbaum [1981] investigate the reconstruction of shape from outline drawings. They note that boundary-defining lines in drawings arise from what they call extremal boundaries (which we shall call smooth occluding boundaries) and discontinuity boundaries (which we shall call sharp occluding boundaries) (Fig. 2).

Boundary lines contain surface normal information. Clearly the line of sight just grazes (is orthogonal to) the surface at a smooth occluding boundary, and the surface normal there is fully determined, being orthogonal both to the line of sight and the boundary contour. At a sharp occluding boundary, the normal is constrained to lie in the plane normal to the (three-dimensional) space curve of the sharp boundary. Object boundaries are an important source of boundary conditions (known orientations) for shape from shading calculations (Section 5.4).

Given a line drawing, Barrow and Tenenbaum propose a process to interpolate surfaces inside boundary contours. First the boundary contours are identified and labelled as smooth or sharp. They are projected into 3-space into the curve minimizing torsion and spatial derivative of curvature, (maximizing planarity and circularity). Thus the surface normals near the boundary contours are known exactly for smooth occluding boundary contours, or restricted to a plane for sharp occluding boundary contours. A linear interpolation in polar space is performed in the interior of a region bounded by the contours, and thus even in the absence of shading a surface is constructed. The scheme has the flavor of a relaxation algorithm because the implementation interpolates using only local information that must propagate around the region.

The surface reconstructed by interpolation is the correct one for the sphere and cylinder, whose normals vary linearly with image displacement. The scheme tries to produce surfaces with uniform mean, principal, and Gaussian curvature. This surface reconstruction without shading information is interesting as a pure case of global relaxation unaffected by shading corrections. The fact that it can work reasonably well by itself makes an approximately cylindrical or spherical surface a poor choice to demonstrate the capabilities of a shape-from-shading system.

5.4 Global Relaxation -- Shape from Shading

Our work is based on the method of Ikeuchi and Horn [1981], which is briefly outlined in this section.

An important starting point for shape from shading calculations is boundary conditions, which in this context are known values of orientation at surface points. Without these, the irradiance equation (3.1) has infinitely many solutions [Bruss 1981]. The constrained values of orientation can arise at the boundary of the object (Section 5.3), or from specular or singular points that indicate an unambiguous orientation. With a Lambertian reflectance function and known light source position, the brightest non-specular points (the singular points) in the image may be taken to have normals in the light source direction. Simple geometric arguments yield the orientation at specular points.

Given adequate boundary conditions, the goal is to find orientations elsewhere on the surface that minimize the sum of two error terms. One term, r_{ij}, measures the disagreement between the intensity at a point I(i,j) (also written I_{ij}) and the intensity predicted by the reflectance map (R(p(i,j),q(i,j)) or R(f(i,j),g(i,j)), say) for the current value of the orientation at that point.

$$r_{ij} = [I_{ij} - R(f_{ij}, g_{ij})]^2 \qquad (5.4.1)$$

The other error term is s_{ij} (eq. 5.4.1), measuring departure from surface smoothness.

The total error e is thus

$$e = \sum_i \sum_j (s_{ij} + w\, r_{ij}) \qquad (5.4.2)$$

where w is a weight determining the relative importance of smoothness and reflectance errors.

Ikeuchi and Horn find an analytic expression for the minimum of the error e, and then construct a Gauss-Seidel like, parallel iterative relaxation algorithm to find orientations minimizing e. The computation can proceed independently in the two orientation parameters (in their case f and g, the stereographic coordinates), and in each *the derivative of the smoothness error term is simply* proportional to the difference of the component and the average of its four neighbors. For f,

$$\partial s/\partial f_{ij} = 2f^{*}_{ij}$$

$$= 2[f_{ij} - (f_{i+1j} + f_{ij+1} + f_{i-1j} + f_{ij-1})/4]. \qquad (5.4.3)$$

The relaxation algorithm computes a new f_{ij} (and g_{ij}) in each iteration; the new component is the average of the four-neighbors in the last iteration, modified by a term expressing the shading constraint. In terms of (f,g) space, the algorithm first initializes f^{0}_{ij} and g^{0}_{ij} to the apriori known orientations, and then on the n+1st iteration,

$$f^{n+1}_{ij} = f^{*n}_{ij} + w[\,I(i,j) - R(f^{*n}_{ij}, g^{*n}_{ij})]\,\partial R/\partial f$$

$$g^{n+1}_{ij} = g^{*n}_{ij} + w[\,I(i,j) - R(f^{*n}_{ij}, g^{*n}_{ij})]\,\partial R/\partial g \qquad (5.4.4)$$

This relaxation algorithm finds surface orientations that minimize a weighted sum of error measures, given an intensity image and a reflectance map as input. The algorithm has many pretty features, and much that is usable in other contexts. It has the limitation that considerable apriori information about the imaging situation must exist.

6. Parameter Transforms

The parameter transform is a generalization of the Hough transform [Duda and Hart 1972; Ballard 1981; Brown 1982]. As such it has well-known pleasant properties of robustness in noise and low computational overhead. It is also of interest as a computational mechanism that is implementable in connectionist architectures [Feldman and Ballard 1981].

Parameter transforms can occur between any two related spaces. Let a parameter (perhaps read off an input parameter image) be a vector (x, a(x)) in a space A, and another parameter (perhaps a feature parameter derived from image space) be a vector b in a space B. Then there is often a physical constraint F that relates a(x) and b so that F(a,b) = 0. Sometimes F may be expressed analytically (as in the Hough transform for parametric curves [Duda and Hart 1972]); sometimes F is best expressed as a relation table (as in the GHough method for nonparametric shape detection [Ballard 1981]).

A particular image may give rise to a set of values {ak} from parameter space A, where ak = a(xk). The set {ak} is only consistent with certain elements in the space B allowed by the relation F. The parameter transform is a way of computing which elements those are and finding the most popular one. For each ak compute the set

$$Bk = \{b \mid ak \text{ and } F(ak,b) < \delta b\}. \qquad (6.1)$$

Bk is the set of elements in space B that are consistent with ak. Define H(b) as the number of times the value element b occurs in the union of all the sets Bk. H(b) is thus a histogram that counts the number of "votes" collected by parameter b on the basis of evidence provided by the ak, and is the output of the parameter tranform from space A to space B for the image described by {ak}. Usually A and B are discrete, and δb relates to the quantization in B. If H is normalized, a normalized confidence measure that the feature with parameter b is present in the image is

$$C(b) = H(b) / \Sigma H(b). \qquad (6.2)$$

7. Implementation

This work assumes an orthographic image projection, with a point illuminant at infinity. With a Lambertian reflectance function, any constant distribution of illumination can be achieved by one point source [Silver 1980; Pentland 1981]. We use the polar space (Gaussian sphere) orientation representation, and have also tried the spherical (θ,φ) representation.

One of our experiments was to derive shape and illuminant direction in parallel, using parameter transforms. Intuitions from photometric stereo (Section 5.1) help in understanding the parameter transform approach to calculating illuminant direction. In photometric stereo, three known illuminant positions allow surface *normal determination. In a dual problem, by reversing the roles of* the vectors, three known surface normals allow determination of the illuminant direction. An image intensity I_{ij} at a point whose surface normal is **n** = (x,y,z) is consistent with a set of illumination directions forming a cone in Cartesian space. The cone's axis is **n**, and the set includes all directions making the correct constant angle with **n**. Rather than follow the analytic dual solution, we use a parameter transform *approach that does not involve simultaneous* equation solution.

In a stage of the illuminant direction calculation, surface points with hypothesized directions each "vote" for the set of illuminant directions with which they are consistent.

The Lambertian reflectivity function means that the irradiance equation is simply

$$I_{ij} = k\ \mathbf{n}_{ij}.\mathbf{L}.$$

That is, the brightness at image location (i,j) is proportional to the cosine of the angle between the normal **n** at (i,j) and the illuminant direction **L**.

In the notation of Section 6, **a** is the intensity and surface normal direction information, **b** is the illuminant direction, and F is the angle condition.

$$\mathbf{a} = [I_{ij}, \mathbf{n}_{ij}]$$

$$\mathbf{b} = [\mathbf{L}]$$

$$F = k\ \mathbf{n}_{ij}\cdot\mathbf{L} - I_{ij} = 0.$$

In a parameter transform implementation, the votes are collected in a discrete accumulator space of orientations. After all the voting, the cell with the most votes is taken to indicate the illuminant direction for the next round of shape calculation. Choosing the direction with the most votes is equivalent to using a mode, rather than a mean or least-squared error regression calculation. The parameter transform thus is unaffected by noise until the noise actually overwhelms the good data. The accumulator space for illumination directions is a geodesic tesselation of the Gaussian sphere [Brown 1979]; this construction partitions the sphere into approximately congruent cells. We index the data structure by treating each subdivided icosahedral face (there are twenty of them, each with N^2 cells) as two triangular arrays, one for cells pointing "up", one for cells pointing "down" (Figure 3).

8. Experimental Results

8.1 Orientation Spaces

Eq. (5.3.4), though derived from least-squares minimization arguments, has a simple geometric interpretation, pointed out in [Ikeuchi and Horn 1981]. This is simply that in the iterative process, the normal vector at the (n+1)st iteration is a weighted vector sum of two other vectors. The first, incorporating the smoothness constraint, is the sum of neighboring normal vectors (of the nth iteration). The second, incorporating the irradiance equation constraint, is a vector in the gradient direction of the reflectance as a function of surface normal. Think of the new surface normal being an average of old neighboring ones (for smoothness), tilted so as to be better in agreement with the known radiance of the image.

The various projections of the gaussian sphere onto the plane *require more or less complex expressions for the gradient, but in* polar orientation space it is easy to show that reflectance functions depending only on the angle of incidence (such as Lambertian) have their gradient in the direction of the illuminant. Let **l** be the unit vector in the direction of the illuminant, **n** be the surface normal unit vector, and i be the angle of incidence. Let R(i) = R1(cos(i)) = R1(**l**·**n**). Then

$$\text{grad}R = (\partial R1/\partial \cos i) = \mathbf{l}.$$

Thus for any reflectance function that is circularly symmetric about the illuminant direction, the shading adjustment in the iteration will simply bend the vector toward or away from the illuminant. We wondered how these geometric intuitions were mirrored in algorithms using different parameterizations of orientation, and whether there were some advantage to using one or another orientation space parameterization.

Some orientation space parameterizations have singularities and infelicities such as modular indices (gradient and (θ,φ) space, for example). The process of "averaging neighbors" is usually taken to operate componentwise. We found that to be painful in (θ,φ) space, and geometrically it seems an approximation to averaging in polar space (i.e. just taking the average of unit vectors in 3-space). Thus even when working in (θ,φ) space we computed vector averages in polar space. With that proviso, the convergence and accuracy of relaxation schemes seemed unaffected by the orientation space parameterization. However, we see little reason ever to work in projections of the gaussian sphere -- polar space seems the natural choice.

8.2 Step Size

The relaxation method dictates a shading adjustment to be performed after the smoothness adjustment. There is always a value (either given or being iteratively adjusted) for the illuminant direction. It thus is always possible to compute the tilt of the normal vector that would satisfy the irradiance equation constraint exactly. We wondered how big an adjustment in the direction of fastest improvement be made.

We tried applying varying amounts of the shading correction (from all of it to about one tenth of it). Unsurprisingly, we found that applying smaller corrections produced results of higher accuracy.

8.3 Reflectance Functions

The nonlinearity of the Lambertian reflectance function (a cosine) means that the computation of surface normal corrections is more sensitive for incidence angles near zero, where the cosine is relatively flat. A small difference in intensity can signal a relatively large difference in surface normal. We conjecture that noise in the intensity image (which for many common noise processes is proportional to the intensity or its square root) could combine with the properties of the Lambertian reflectance function to distort the derived shape. To test this hypothesis, we constructed images arising from the reflectance function that is linear in the incidence angle. This reflectance function performs identically to the Lambertian in noiseless conditions. Its performance in noisy conditions is a matter for future research, and is mildly interesting because the Lambertian reflectance function theoretically has higher noise resistance for low angles of incidence but lower noise resistance for high angles of incidence.

8.4 Constraint Interactions

We explored how the surface smoothness constraint interacted with apriori knowledge and boundary constraint.

Surface:	$f(x,y) = \sin(x)$, $(-1.9 <= x <= 1.9)$.
Resolution:	f defined on 33 x 33 grid
Orientation Parameters:	polar
Smoothness Constraint:	average four-neighbors
Irradiance Constraint:	.2(entire shading correction)
Apriori Knowledge:	1. orientations of entire boundary known 2. orientation known on opposite sides (two cases)
Boundary Constraint:	initial orientations fixed, unknown free.

The surface has a maximum and minimum where the surface normal changes sense in the x direction: all normals are in the x-z plane. The results of these experiments are shown in Figures 4 and 5.

Figure 4a shows the input image (the illuminant is at the viewpoint). When normals are known apriori for only the top and bottom boundary (Fig. 4b), the normals are reconstructed immediately (Fig. 4c) with no appreciable error. When the apriori knowledge is of the left and right boundaries, the result (not illustrated) is disastrous, with the smoothness constraint preventing the normals from changing direction across the peak and trough. The resulting surface fits the irradiance constraint well, but has serious and permanent normal direction errors.

Figure 5 shows the case that normals are known apriori around the entire boundary (Fig. 5a). After some time, an intermediate stage is reached at which conflicting normal information has propagated in from the sides and met along the diagonals. Here the averaging process yields equivocal results, and the surface has severe infelicities in both smoothness and irradiance (Fig. 5b,c). After a time the smoothness constraint has won out, deriving the expected surface (Fig 5d, e, f). The irradiance constraint weight was important here: with the irradiance constraint set to add 100% of the shading correction, a stable point is reached at the situation of 5b and c. This equilibrium is fragile: when at one point a program bug introduced a minor assymetry, convergence to the "expected" surface occurred.

8.5 Boundary, Illuminant, and Apriori Constraint

We wanted to explore the interaction of boundary constraint, illuminant direction constraint, and apriori knowledge of boundary orientation. To that end we ran a series of six experiments. For each we used the following setup.

Surface:	$f(x,y) = \cos(xy)$, $(-2.1 <= x,y <= 2.1)$
Resolution:	f defined on 33 x 33 grid
Orientation Parameters:	polar
Smoothness Constraint:	average four-neighbors
Irradiance Constraint:	approx. .2(entire shading correction)
Hough Space:	order 7 icosahedral geodesic tesselation (49*20 cells)
Hough Voting Scheme:	one vote per cell
Illuminant Direction:	known or unknown (two cases)
Apriori Constraint:	orientations known or unknown (two cases)
Boundary Constraint:	known orientations fixed or orientations constrained to lie in plane (two cases)

Some explanation:

The surface is not radially symmetric, and has enough variation in its normals to provide a challenge to the shape-from shading algorithm. As set forth in Section 7, we use a Hough transform technique to attempt to derive illuminant direction in parallel with the shape determination. With this technique, used in the unknown illuminant cases, some of the already-computed surface normals "vote" for the illuminant direction, in a dual of the photometric stereo process. In Fig. 1, we can imagine each ellipse being a pattern of votes: where the three ellipses intersect, there will be a peak of votes, and that is the desired normal.

Similarly, in the illuminant direction computation, each normal casts a circle of votes onto the gaussian sphere, denoting all illuminant directions at the angle with the normal that produces the known intensity. There are many variations in how such votes can be cast. In terms of [Brown and Curtiss 1982], this scheme is "jagged" (not anti-aliased, one vote per cell). We have also implemented a CHough scheme [Brown 1982]; in noiseless data it does not improve the final results.

The orientations of normals around the border are initially taken either to be known exactly or to be smoothly related to each other but not correctly known. In the latter case the corners are left in the correct orientation and as x and y approach 0 the normals are increasingly rotated in the plane normal to the boundary curve. The rotation increases by 3 degrees,per boundary element linearly toward a maximum of 45 degrees.

The boundary constraint is either that the boundary normals are fixed and correct or that they are constrained to rotate in a plane normal to the boundary curve. The former constraint is the "smooth occluding contour" constraint [Ikeuchi and Horn 1981], the latter the "sharp occluding contour" constraint [Barrow and Tenenbaum 1981]. We did not run the case that the orientations are incorrect but fixed (hence only six cases).

The cases in which the illuminant is known are comparable to the usual shape-from-shading results (e.g. [Strat 1979, Ikeuchi and Horn 1981]). A case in which the boundary is known and is immediately used to calculate the illuminant direction is a Hough implementation of solving the dual photometric stereo equations. It is not interesting except as a test that the votes go to the right places. After such a start, the situation is the same as if the illuminant were known apriori.

Figure 6 shows the original surface.

Figure 7(a-g) shows Case 1, approximating the usual shape-from shading case with smooth occluding boundary, in which illuminant direction is known and boundary normals are known and fixed. In these runs, illuminant is at θ = 19 or 20 degrees, and [phi] is variously at 45, 90, and 135 degrees. 7a-d show the initial, correct, and computed normals and the normal differences. 7e shows the input image, 7f shows the image that would result from the reconstructed surface, and 7g shows the reconstructed surface integrated from its surface normals. In all the cases, the irradiance errors are negligable, so the image-like figures will not be repeated. The shading term is .2(complete correction). The computed normals are not perfect, being at their worst where they have to change sign in x or y. This could result from the "looseness" of the sun in these combined calculations, where illuminant position is influenced by the surface calculations.

Fig. 8(a - d) shows Case 2, in which the illuminant direction and boundary normals are both known correctly, but the normals are free to rotate in the plane normal to the boundary curve (the sharp occluding boundary case). The shading term is .3(complete correction). Fig. 8a is initial normals, 8b,c show normals after 25 iterations and in their final position. 8d shows the normal errors. The normal convergence is often poor until the normals propagating in from the boundaries meet in the center. This might be alleviated by increasing the shading weight to overcome the tendency of normals to stay at the slant of the boundary normals.

Figure 9(a-d) show Case 2: illuminant direction is known, initial normals incorrect and free to rotate in the plane orthogonal to the boundary curve. Shading term is .3(complete correction). The initial boundary normals are smoothly related, but twisted as they get farther from the corners (compare 9a to 8a). 9b,c, and d show the correct normals, computed normals, and the normal errors. The computed normals are fairly accurate, and again the irradiance error is negligable, showing that shading corrections can overcome bad boundary information.

Figure 10(a-d) shows Case 4: illuminant direction unknown, boundary normals initially correct and fixed. This is close to a Hough implementation of the photometric stereo dual. It is not quite, because the initially known normals are not allowed to vote for illuminant direction, but the inner rings obtained by averaging are. The illuminant is at (19, 90), initilly guessed at (0,0), i.e. overhead, or in the viewing direction. Shading term is .2(complete correction). 10a and b show the computed normals and errors. 10c shows the Hough accumulator array with votes for sun position, and 10d the reconstructed surface.

Figure 11(a-e) shows Case 5: illuminant direction unknown, boundary normals initially correct but free to rotate in the plane orthogonal to the boundary curve. Here again the initially correct boundary normals helped the Hough process find the correct illuminant direction immediately. Shading term is .2(complete correction). 11a and b show the computed normals and errors after 21 iterations. 11c and d show two displays of the accumulator array for illuminant direction, and 11e shows the reconstructed surface.

Figure 12(a-e) shows Case 6: illuminant direction unknown, boundary normals initially incorrect and free to rotate in the plane orthogonal to the boundary curve. Here the boundary normals were twisted by up to 45 degrees as described above. The illuminant was at (19, 135), and guessed initially to be at (0,0). Shading term is .2(complete correction). 12a and b give the computed normals and differences, showing significant errors. The initial Hough operation (shown in 12c and d) does not yield a sharp peak for the illuminant direction (this is no surprise). The derived direction was between the correct direction and the first guess. The next iteration of illuminant calculation (12.e) in fact found the correct accumulator bin, and after two more iterations settled down to within .05 radian of the correct illuminant direction.

9. Conclusions and Future Work

The main conclusions we draw are the following.

1. Iterative relaxation schemes for shape-from-shading calculations are robust over several design choices.

2. The Gaussian sphere (polar space) parameterization of orientation seems natural, adequate, and elegant.

3. Iterative schemes may suffer from significant anomalies resulting from discrete approximations to continuous quantities.

4. There are several potentially important ideas for improving performance of iterative relaxation schemes (see below).

5. It is possible to use Hough methods to derive illuminant direction in parallel with shape-from-shading calculations only so long as the normals are fairly well known. Otherwise their voting behavior is too incoherent or wrong to be overcome by smoothness constraints, and the derived surface is distorted.

Thanks to this work and to that of others, we have several ideas for improving the performance of these algorithms.

1. Use of local information. The work of Pentland [1982] shows how information about slant, tilt, and illuminant direction may be inferred immediately from local measures of intensity differentials. Especially when the image is not noisy, such information can provide both apriori information and constraint for parallel-iterative schemes. In fact, quite reasonable shape results may be derived strictly with local clues, using no global relaxation. Pentland's version of our derivation of shape and illuminant direction in parallel is his "bootstrapping" method of iterative successive computation of illuminant direction and shape.

2. The issue of smoothness constraint is an interesting one. The one we used (that of Ikeuchi and Horn [1981]) is best satisfied by a flat surface. In surfaces with substantial difference in principal curvatures at a point, (especially surfaces with zero or negative gaussian curvature), it may be useful to try to decouple the smoothness measure in the two principal directions. One approximation to this is still to enforce smoothness through averaging, but only along iso-brightness contours. This accomplishes the decoupling for cylindrical surfaces, though not exactly for hyperbolic surfaces.

3. Performance in noisy conditions was investigated by Strat [1979], and similarly we would like to study performance of the Hough scheme with various reflectance functions in noisy conditions. Also of interest is the CHough scheme, which sharpens peaks in parameter space [Brown 1982b].

4. This work is one of several applications we should like to implement with a cache-based Hough Transform [Brown and Sher 1982].

REFERENCES

Ballard, D.H. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition* 13, 2, 111 - 122, 1981.

Ballard, D.H. and O.A. Kimball. Rigid body motion from depth and optical flow, TR 70, Computer Science Department, Univ. of Rochester, Dec. 1981. (submitted to *CGIP* special issue on computer vision).

Barrow, H.G. and J.M. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, A.R. Hanson and E.M. Riseman (eds), Academic Press, New York 1978.

Barrow, H.G. and J.M. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence* 17, 75-116, 1981.

Brooks, M.J. Surface normals from closed paths. *Proc.* Sixth IJCAI, Tokyo Japan, 98-101, 1979.

Brown, C.M. Two descriptions and a two-sample test for 3-D vector data. TR 49, Computer Science Department, Univ. of Rochester, Feb. 1979.

Brown, C.M. Bias and noise in the Hough transform I: theory. TR 105, Computer Science Department, Univ. of Rochester, May 1982.

Brown, C.M., and M. Curtiss. Bias and noise in the Hough transform II: experiments. TR113, Computer Science Department, Univ. of Rochester, August 1982.

Bruss, A.R. The image irradiance equation: its solution and application. PhD. thesis and MIT AI Memo 623, June 1981.

Cook, R.L., and K.E. Torrance. A reflectance model for computer graphics. *Computer Graphics* 15, 3 (*proc.* SIGGRAPH 81) 307-315, August 1981.

Duda, R.O. and P.E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Comm. ACM* 15, 1, 11-15, Jan. 1972.

Horn, B.K.P. Shape from shading: a method for obtaining the shape of a smooth opaque object from one view. Tech. Report MAC-TR-79, Project MAC, MIT, 1970.

Horn, B.K.P. Obtaining shape from shading information, in Winston, P.H. (Ed.), *Psychology of Computer Vision*, McGraw-Hill, NY 115-155, 1975.

Horn, B.K.P. Understanding image irradiances. *Artificial Intelligence* 8, 201-231, 1977.

Horn, B.K.P. and R.W. Sjoberg. Calculating the reflectance map. *Applied Optics* 18, 11, 1770-1779.

Huffman, D.A. Surface curvature and applications of the dual representation. In *Computer Vision Systems*, A.R. Hanson and E.M. Riseman (eds), Academic Press, New York 1978.

Gauss, K.F. General investigations of curved surfaces (1827), Translated by A. Hiltbeitel and J. Morehead. Raven Press, Hewlett, N.Y. 1965.

Gilchrist, A.L. The perception of surface blacks and whites. *Scientific American* 240, 3, 112-124, March 1979.

Grimson, W.E.L. A computational theory of visual surface interpolation. MIT AI Memo 613, June 1981.

Ikeuchi, K. Determination of surface orientations of specular surfaces by using the photometric stereo method. *IEEE-PAMI*, to appear.

Ikeuchi, K. and B.K.P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence* 17, 141-184, 1981.

Mackworth, A.K., Interpreting pictures of polyhedral scenes, *Artificial Intelligence* 4, 121-137, 1973.

Nicodemus, F.E., J.C. Richmond, J.J. Hsia, I.W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. NBS monograph 160, U.S. Dept. of Commerce, NBS, 1977.

Pentland, A. P. Local computation of shape, PhD thesis, MIT, 1982a.

Pentland, A. P. Local computation of shape. Technical Memo, SRI International, preliminary version, 1982b.

Silver, W. Determining shape and reflectance using multiple images. S.M. thesis, Dept. of EE and CS, MIT, 1980.

Strat, T.M. A numerical method for shape from shading from a single image. S.M thesis, Dept. of EE and CS, MIT, 1979.

Whitted, T. An improved illuminations model for shaded display. *Comm. ACM* 23, 6, 343-349, June 1980.

Woodham, R.J. Reflectance map techniques for analyzing surface defects in metal castings. PhD thesis and MIT AI Memo 457, June 1978a.

Woodham, R.J. Photometric stereo, MIT AI Memo 479, June 1978b.

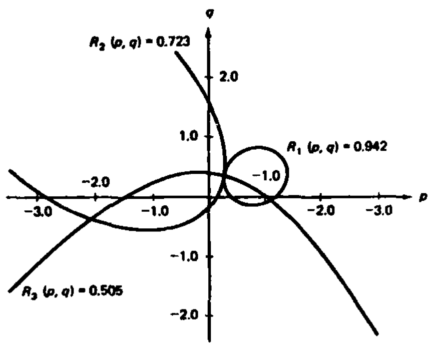


Figure 1: Three light source positions have three associated reflectance maps. In each situation, an image point (i,j) has a brightness that can be explained by a locus of orientations (here, points in p,q space). The intersection of these loci is the orientation of the surface at the point.

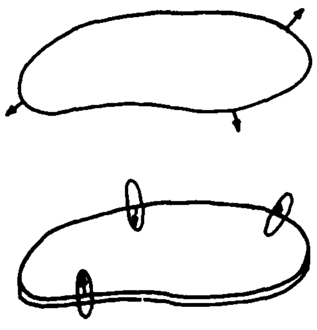


Figure 2: (a) The solid, smooth, potato-like object has a smooth occluding boundary, at which the normal is uniquely determined. (b) The normal near the boundary of the potato-chip-like object has a sharp occluding boundary, at which the normal is constrained to lie in a plane.

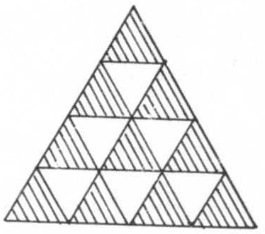


Figure 3: A subdivided icosahedral face may be indexed as two triangular arrays, for cells pointing "up" (shaded) and down.

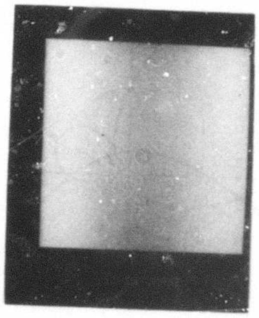


Figure 4a

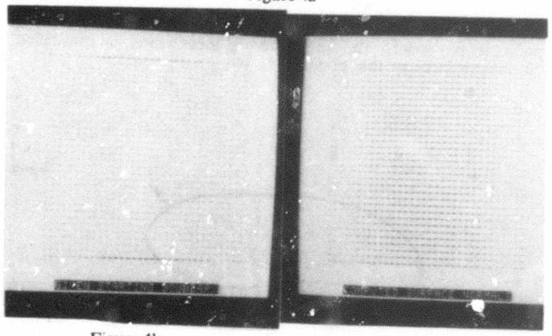


Figure 4b

Figure 4c

Figure 4: A sine surface with partial apriori normal information (see text).

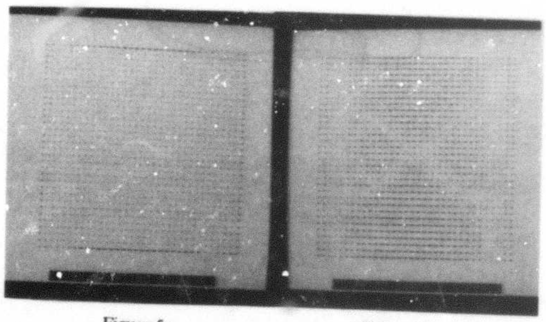


Figure 5a

Figure 5b

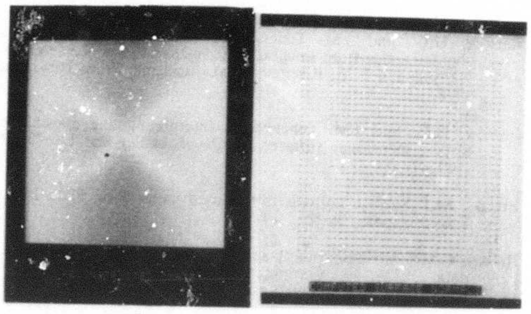


Figure 5c

Figure 5d

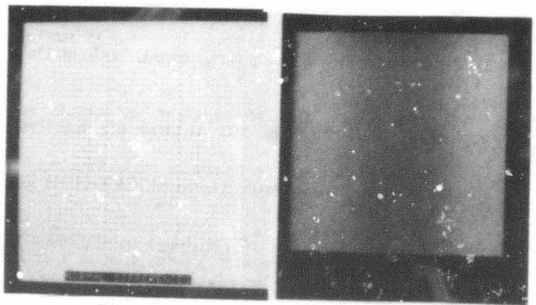


Figure 5e

Figure 5f

Figure 5: The surface of Fig. 4 with complete apriori normal information (see text).

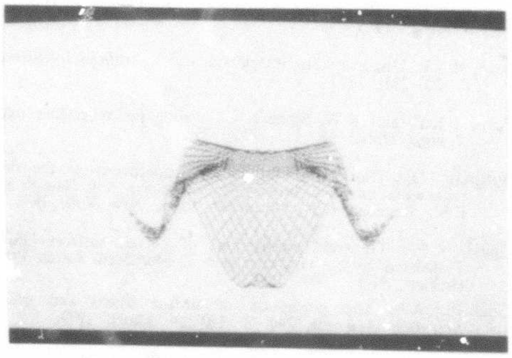


Figure 6: The surface f = cos(xy).

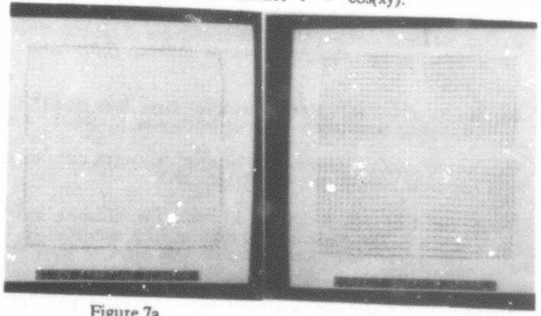


Figure 7a

Figure 7b

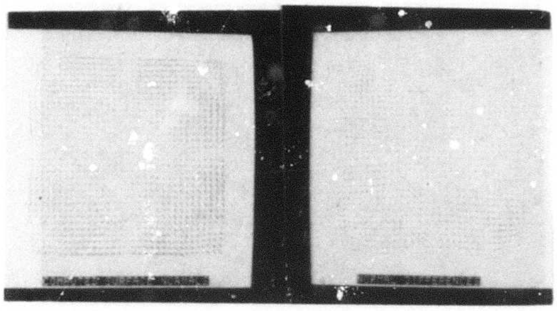


Figure 7c Figure 7d

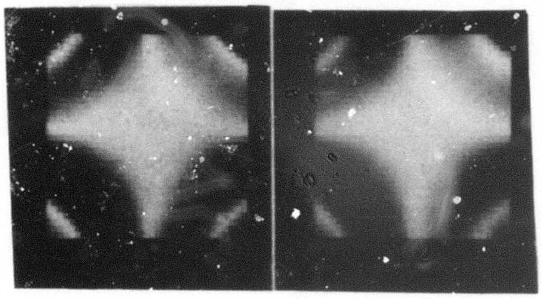


Figure 7e Figure 7f

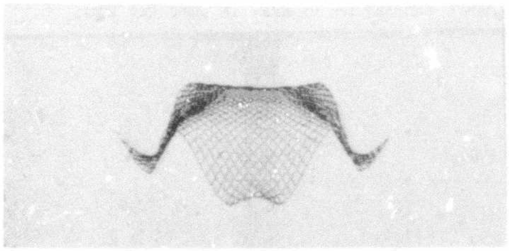


Figure 7g

Figure 7(a-g): Case 1: apriori known illumininant, initially correct and fixed boundary normals (see text).

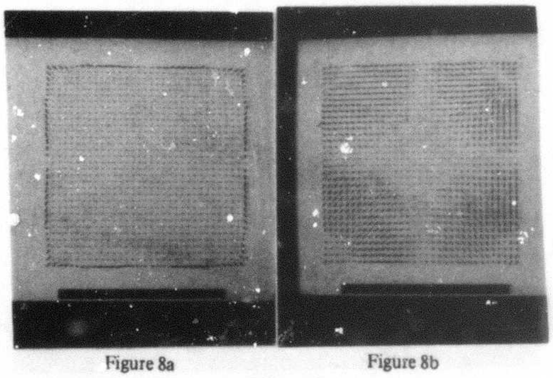


Figure 8a Figure 8b

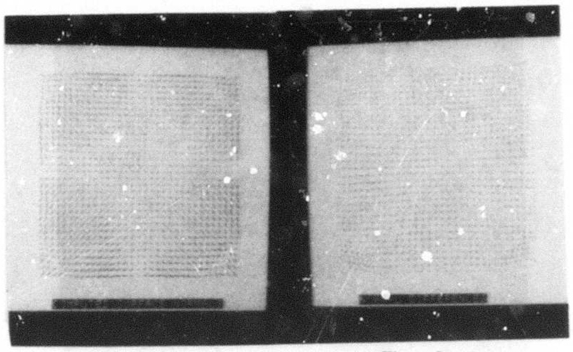


Figure 8c Figure 8d

Figure 8(a-d): Case 2: apriori known illuminant, initially correct boundary normals free to rotate in plane (see text).

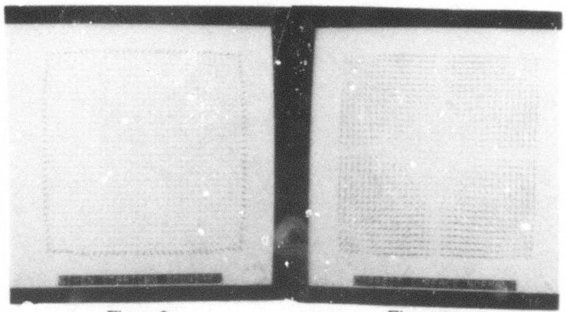


Figure 9a Figure 9b

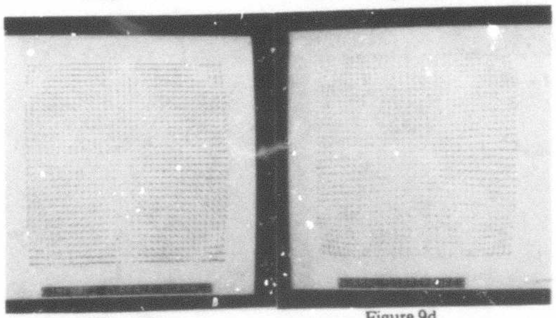


Figure 9c Figure 9d

Figure 9(a-d): Case 3: apriori known illuminant, initially incorrect boundary normals free to rotate in plane (see text).

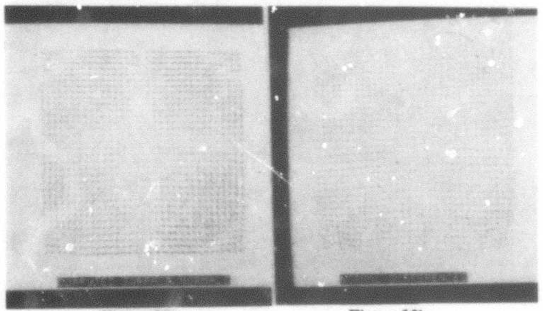


Figure 10a Figure 10b

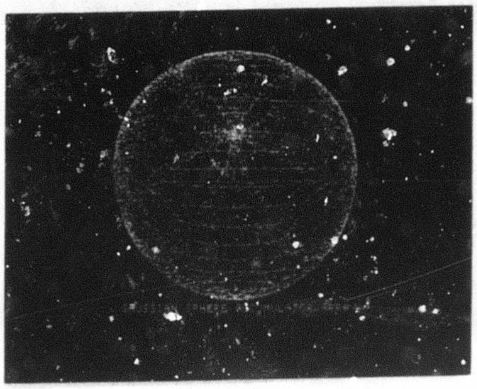


Figure 10c

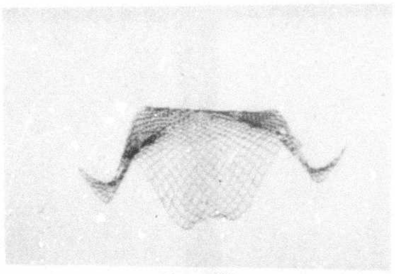


Figure 10d

Figure 10 (a-d): Case 4: unknown illuminant, initially correct and fi:ed boundary normals (see text).

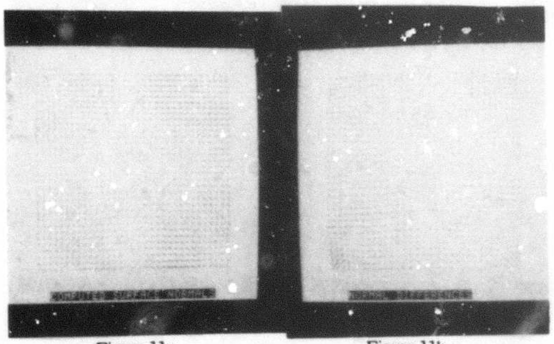


Figure 11a Figure 11b

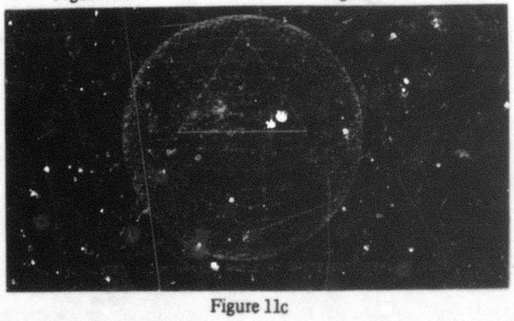


Figure 11c

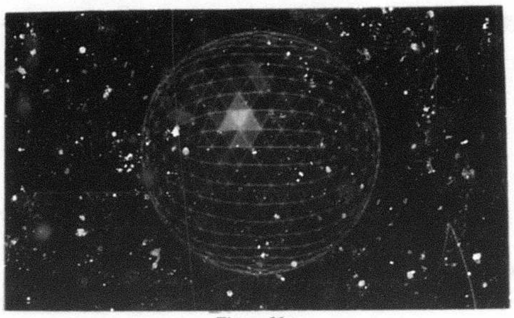


Figure 11d

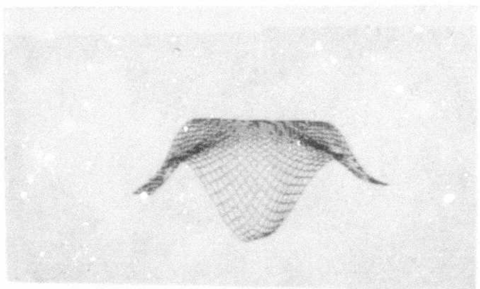


Figure 11e

Figure 11(a-e): Case 5: unknown illuminant, initially correct boundary normals free to rotate in plane (see text).

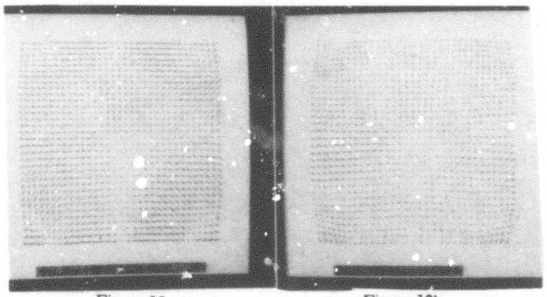


Figure 12a Figure 12b

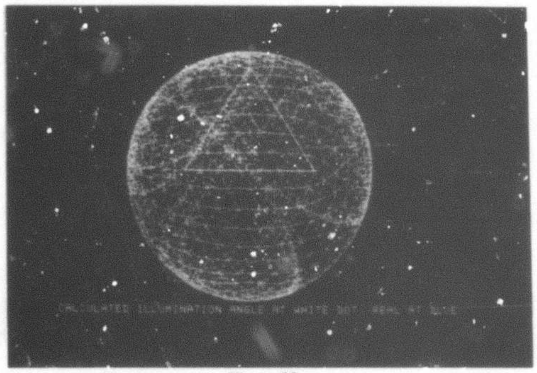


Figure 12c

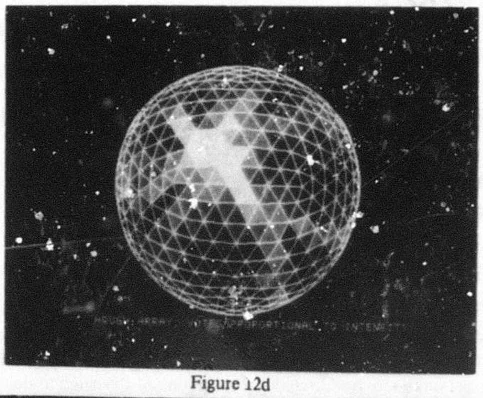


Figure 12d

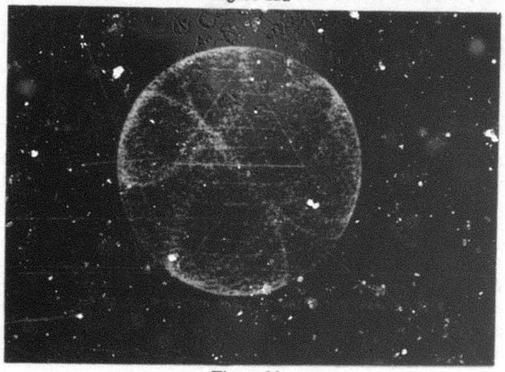


Figure 12e

Figure 12(a-e): Case 6: unknown illuminant, initially incorrect boundary normals free to rotate in plane (see text)

APPENDIX

Invariant Image under Varying Illuminant Direction: The Unconstrained One-Dimensional Case

For this exercise, assume the brightness at a point is some function of the incidence angle (a generalization of Lambertian reflectance). The viewer is at infinity on the +y axis (orthogonal projection) and a point illuminant is at infinity in the +y halfplane. There are no boundary conditions: the slope of the 1-D "surface" is nowhere constrained.

The 1-D "surface" is a function

$$y = f(x).$$

For a given illumination angle, the brightness of the surface at any position x is determined by its slope there. (Fig. A.1).

If the illumination angle were to change by $-\delta$, then the same image would result if all the slopes changed by $-\delta$ to compensate, determining a new surface g(x). Since in the original surface f(x),

tangent angle of f(x) at x $= \gamma$,

and

$$f'(x) = df/dx = \tan\gamma.$$

In the compensating, tilted surface g(x),

tangent angle of g(x) at x $= \gamma - \delta = \beta$,

and

$$g'(x) = dg/dx = \tan\beta = (\gamma + \tan\delta) / (1 - \gamma\tan\delta).$$

Thus

$$g(x) = \int f'(x) / (1 - f'(x)\tan\delta)\, dx + \int \tan\delta / (1 - f'(x)\tan\delta)\, dx.$$

For example, if

$$f(x) = x2,$$

then

$$g(x) = [\ -x/\tan\delta - (1/2)\csc^2\delta(\log(1-2x\tan\delta))].$$

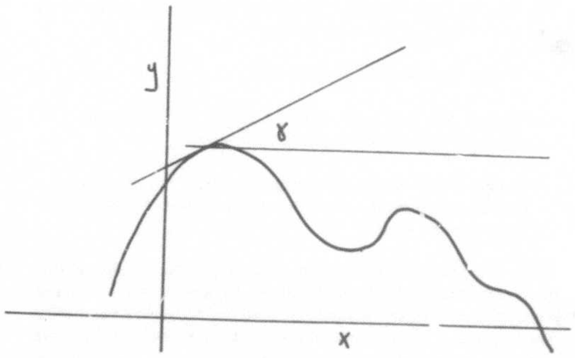


Figure A.1: Imaging geometry for unconstrained 2-D situation.

AD-P000 115

Using Shadows in Finding Surface Orientations

23 June 1982

Steven A. Shafer

Takeo Kanade

Computer Science Department

Carnegie-Mellon University

Abstract

Given a line drawing from an image with shadow regions identified, the shapes of the shadows can be used to generate constraints on the orientations of the surfaces involved. This paper describes the theory which governs those constraints under orthography.

A "Basic Shadow Problem" is first posed, in which there is a single light source, and a single surface casts a shadow on another (background) surface. There are six parameters to determine: the orientation (2 parameters) for each surface, and the direction of the vector (2 parameters) pointing at the light source. If some set of 3 of these are given in advance, the remaining 3 can then be determined geometrically. The solution method consists of identifying "illumination surfaces" consisting of illumination vectors, assigning Huffman-Clowes line labels to their edges, and applying the corresponding constraints in gradient space.

The analysis is extended to shadows cast by polyhedra and curved surfaces. In both cases, the constraints provided by shadows can be analyzed in a manner analogous to the Basic Shadow Problem. When the shadow falls upon a polyhedron or curved surface, similar techniques apply. The consequences of varying the position and number of light sources are also discussed. Finally, some methods are presented for combining shadow geometry with other gradient space techniques for 3D shape inference.

1. Introduction

1.1 The Shadow Geometry Problem

When shadows are present in an image, they provide information which can be used for determining the 3D shapes and orientations of the objects in the scene. The interpretation of shadows in an image involves three distinct processes:

- Finding shadow regions in the image
- Solving the correspondence problem to determine which object has cast each shadow region
- Geometrically deducing information about the objects and surfaces involved on the basis of the identified object/shadow pairs

Techniques for the first task, *finding shadow regions*, have been proposed by many researchers, usually by looking for regions of low intensity with approximately the same hue as some neighboring region [10, 12]. Lowe and Binford [7] and Witkin [17] have proposed criteria which should be satisfied by edges of shadow regions; these can be used to suggest or try to confirm the hypothesis that a particular region is a shadow. Waltz [15] developed a method for labeling lines in line drawings as shadow edges, based on local geometric criteria at vertices.

The *correspondence problem* has been explored primarily by Lowe and Binford [7]. They describe several properties of this correspondence, and include descriptions of the special points of view from which degenerate cases arise. O'Gorman [11] proposed a heuristic method for finding correspondences in the blocks world under orthography.

Geometric interpretation of shadows is also performed by Lowe and Binford [7], who use shadows to determine height in overhead views of airplanes. They measure the distance in the image between the outline of an object and the outline of its shadow, and use similar triangles to conclude that this distance is proportional to the height of the object's edge above the ground. These techniques have been employed in manual photo-interpretation of aerial photographs as well [14].

Waltz [15] used shadows to classify surfaces into several orientation categories depending upon the geometry of the shadows in a line drawing. His categories were qualitative, such as "front left" for an approximately vertical surface tipped to the left.

This paper presents a theory describing the constraints that shadows provide between surface orientations in line drawings, using shadow and surface outlines under orthographic projection. This can be thought of as a method for achieving the same kind of results as Waltz, but computing exact surface orientations rather than simply categorizing the surfaces into classes with similar orientations. The theory presented here subsumes the "shadow-plane" idea suggested by Mackworth [8] as a means for generating gradient-space constraints from shadows.

Shadows cast by and upon curved surfaces have been described by Witkin [16], who derived equations relating surface curvature to curvature of shadow edges in the image. The presentation in this paper is somewhat different, discussing surface *gradient* (local orientation) rather than *curvature* (rate of change of orientation).

1.2 Gradient Space and Line Labeling

This section presents an introduction to the gradient space and line labeling for readers who are not already familiar with these topics.

The coordinate system used in this paper is illustrated in figure 1. The x and y axes are aligned on the image plane in the horizontal and vertical directions, respectively; the z axis points towards the viewer (or camera).

In this paper, it will be presumed that the point (x,y,z) in the scene corresponds to the point (x,y) in the image: this is *orthography*. *Perspective* projection is not discussed in detail in this paper.

Surface orientations can be represented by points in a plane called the *gradient space* (figure 2) [4]. If a surface is represented

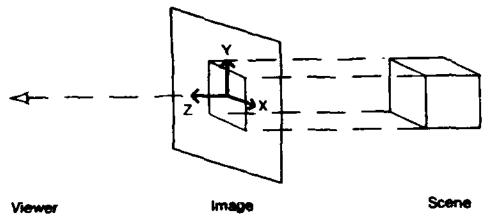


Figure 1: The X-Y-Z Coordinate System

by the equation

$$-z = f(x, y)$$

then its *gradient* is represented by the point:

$$(p, q) = (\partial f/\partial x, \partial f/\partial y)$$

This assigns a natural interpretation to points in gradient space: a surface which is "tipped" to the right is represented to a point on the right side of the origin; a surface tipped left has a gradient to the left of the origin. Similarly, a surface which is tipped up (or down) has its gradient above (or below) the origin. In figure 2, the gradients G_A (etc.) are shown for the surfaces S_A (etc.) in the line drawing at the right.

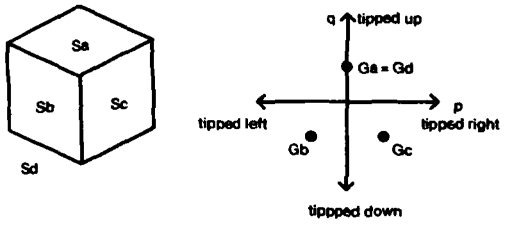


Figure 2: The Gradient Space

Before computing surface orientations, it is common to attempt to produce a *line drawing* from an image, in which all the surfaces are outlined. Huffman and Clowes [4, 2] showed that the edges (line segments) in a line drawing do not all represent the same three-dimensional surface configuration. The four types of edges they discovered are shown in figure 3, along with the half-planes containing the surfaces which meet at each type of edge. At a *convex* edge, the surfaces recede from the viewer as you travel farther from the edge. At a *concave* edge, the surfaces approach the viewer as you travel farther from the edge. At an occluding edge, only one of the two surfaces involved is directly visible in the image. Waltz [15] developed an algorithm for assigning these labels to the edges in a line drawing.

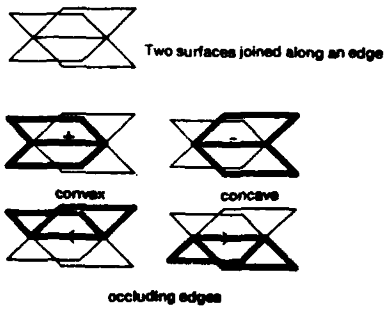


Figure 3: Line Labels and Surface Intersections

The *convex* and *concave* labels indicate relationships between the gradients of the surfaces which meet along an edge [8]. When two surfaces are joined along a convex edge, their gradients lie along a line in gradient space which is perpendicular to the edge in the image (figure 4). Furthermore, the relative positions of the surface gradients will be the same as the relative positions of the surfaces in the image. When two surfaces meet at a concave line, the gradients are still on a perpendicular line in gradient space, but the relative positions are reversed.

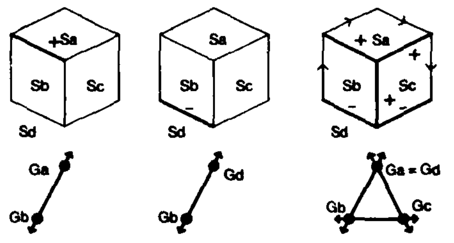


Figure 4: Line Labels and Gradient Space Relationships

In general, if an edge $E = (\Delta x, \Delta y)$ is contained on a surface with gradient $G = (p, q)$, then the edge corresponds to the three-dimensional vector $(\Delta x, \Delta y, \Delta z)$ where

$$-\Delta z = G \cdot E \tag{1.1}$$

In this paper, a method is proposed for assigning Huffman-Clowes line labels to shadow-making edges and shadow edges in a line drawing, and for using the resulting gradient space relationships to determine surface orientations.

2. The Basic Shadow Problem

The *Basic Shadow Problem* is:

> Given a line drawing such as Figure 5, what constraints exist between the occluding surface S_O and the shaded surface S_S?

For simplicity, we will begin by assuming that the surfaces are both flat, and that orthographic projection is used. We will also, for the time being, presume that the light source is infinitely far away; this means that all *illumination vectors* (light rays emanating from the light source) are parallel.

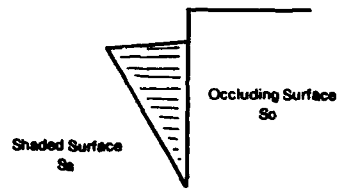


Figure 5: The Basic Shadow Problem

2.1 Solution of the Problem

To show the proper correspondences, the edges and vertices can be labeled as in figure 6, where edge E_{S1} is the shadow edge corresponding to E_{O1}, E_{S2} is the shadow of E_{O2}, and vertex V_{S12} is the shadow of V_{O12}.

Consider the physical interpretation of edge E_{S1}. Some light rays just graze past S_O at E_{O1}, and continue on to strike S_S along E_{S1}. This set of rays form a surface (a piece of a plane), in fact the plane

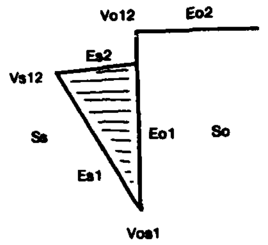


Figure 6: Basic Shadow Problem -- Correspondences Labeled

containing E_{O1} and E_{S1}. This is a surface consisting of "illumination vectors": call it surface S_{I1} (Figure 7).

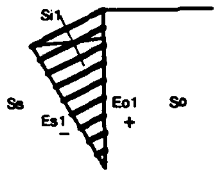


Figure 7: Basic Shadow Problem -- Illumination Surface 1

Suppose we were to cut a piece of cardboard and fit it into the space occupied by S_{I1}. Then, this cardboard and S_O would be joined along E_{O1}, a *convex* edge. Using Huffman-Clowes line labeling [4], this edge can be given the label +. Similarly, E_{S1} joins S_S and S_{I1}, and is *concave*; it receives the label −.

As Mackworth showed [8], these line labels can be mapped into constraints in the gradient space. The gradient of S_O (G_O) and the gradient of S_{I1} (G_{I1}) must be joined by a line perpendicular to E_{O1}; since the label of E_{O1} is +, G_O and G_{I1} have the same relative positions as S_O and S_{I1}. Similarly, G_{I1} and G_S are joined by a line perpendicular to E_{S1}, with relative positions reversed because of the − label. These facts yield the relationship shown in figure 8 in the gradient space. However, we do not yet know the position of this figure in gradient space, nor the distances involved: only the angles are known.

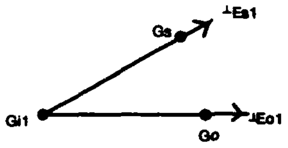


Figure 8: Gradient Space Constraints from Illumination Surface 1

S_{I1} is not the only illumination surface in the Basic Shadow Problem: the illumination surface S_{I2} joins edges E_{O2} and E_{S2} (Figure 9). Along E_{S2}, the − label is assigned; along E_{O2}, the − label refers to the junction of S_O and the upper half-plane of S_{I2}. The gradient space constraints are shown in figure 10. Note that it is possible for E_{O2} and E_{S2} to be parallel, in which case the two rays shown in gradient space are coincident.

A third constraint in the gradient space arises from the fact that an edge E_{I1} can be drawn joining V_{O12} and V_{S12} (Figure 11). This edge lies in a line which passes through the light source, since V_{S12} is the shadow of V_{O12}. The vector I pointing at the light source can be represented in gradient space by a point G_I, which is the gradient of those surfaces whose normal vectors are parallel to I. Since E_{I1} lies in the projection of this vector onto the image plane, the point G_I must lie along a line in gradient space, passing through

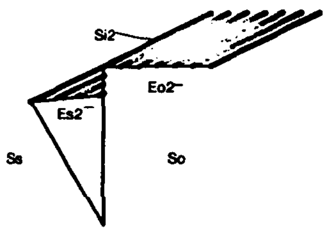


Figure 9: Basic Shadow Problem -- Illumination Surface 2

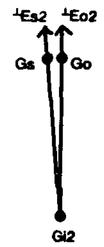


Figure 10: Gradient Space Constraints From Illum. Sur

the origin, and parallel to E_{I1} (Figure 12). It is not known, hov how far this point G_I is from the origin; suppose this is determined somehow (as described below), and call the distance k. It should be noted that k represents the relative change in z with a change in x or y along the illumination vector. It is defined by this equation:

$$k = \mathrm{sqrt}(\Delta x^2 + \Delta y^2) / \Delta z = \|E_{I1}\| / \Delta z \qquad (2.1)$$

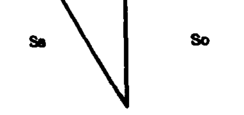


Figure 11: Basic Shadow Problem -- Illumination Vector

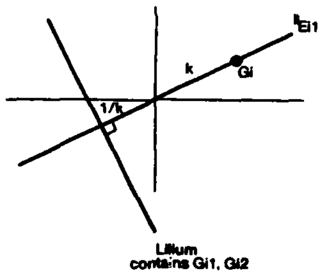


Figure 12: Gradient Space Constraints From Illumination Vector

The line L_{illum} perpendicular to E_{I1}, and located at a distance $1/k$ from the origin, represents the locus of the gradients of all planes which contain the illumination vector I. This is the set of all illumination planes, and in particular contains both S_{I1} and S_{I2}; thus, G_{I1} and G_{I2} are points on the line L_{illum}. This property subsumes the property of G_{I1} and G_{I2} that they must be joined by a line perpendicular to E_{I1}, since E_{I1} can be given the label + or − (depending on which half-planes the line label refers to).

The line L_{illum} is the same as the *terminator* described by Horn in [3]. *It separates the gradient space into two half-planes: the half-*plane containing G_i represents the gradients of all planes that will receive illumination, while the other half-plane contains the gradients of *self-shadowed* surfaces (facing away from the light source).

This is the extent of the information available from the line drawing in figure 5. Since each gradient is an ordered pair (p, q), the problem has six parameters to be computed:

- *(2 parameters)* G_O, the gradient of S_O
- (2 parameters) G_S, the gradient of S_S
- (2 parameters) G_i, the direction of the light source.

From the Basic Shadow Problem geometry, three constraints are provided:

- The angle G_O-G_{i1}-G_S, which comes from the angle E_{O1}-E_{S1}
- The angle G_O-G_{i2}-G_S, which comes from the angle between E_{O2} and E_{S2}
- The direction of the line L_{illum} (containing G_{i1} and G_{i2}), which comes from the direction of E_{i1}.

We would therefore expect that three parameters must be given in advance, and the other three can be computed from the geometry.

Let us suppose, for example, that the value k is given (*the relative depth component of the direction of the light source), and that* G_S is known (the relative orientation of the background with respect to the camera). The construction in the gradient space for computing G_O proceeds as follows (Figure 13).

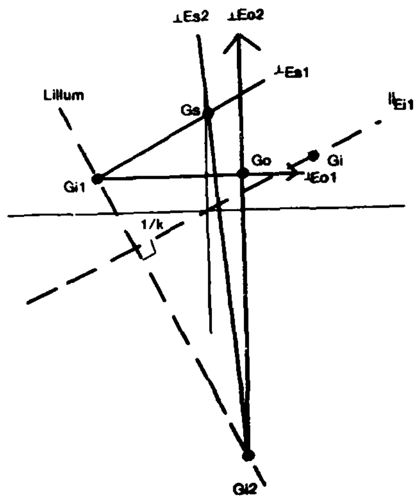


Figure 13: Solution to Basic Shadow Problem

1. Draw the line parallel to E_{i1} through the origin. Since k is known, G_i and L_{illum} can be found.
2. Plot G_S, which was given. Through this point, draw a line perpendicular to E_{S1}. Where it intersects L_{illum} must be G_{i1}. Through G_{i1}, draw a line perpendicular to E_{O1}. G_O must lie on this line.
3. From G_S, draw a line perpendicular to E_{S2}. Where it intersects L_{illum} will be G_{i2}. From there, draw a line perpendicular to E_{O2}. Since G_O must lie on this line, the intersection of this line with the final line from step (2) above must be G_O.

In [13], the solution for the Basic Shadow Problem is shown to be of the form:

$$G_o = \begin{bmatrix} p_O \\ q_O \end{bmatrix} = \begin{bmatrix} Q\,R\,S \\ T\,U\,V \end{bmatrix} \begin{bmatrix} p_S \\ q_S \\ 1/k \end{bmatrix}$$

where Q, R, S, T, U, and V can be computed from measurements of the line drawing (or image).

2.2 Relationships Among the Parameters Supplied in Advance

In the example above, G_S and k were needed before the *construction could take place. In practice,* a program for a specific application may not be able to compute these particular parameters.

It is possible to begin the construction with any three of the six pieces of information specified in advance, as long as none are redundant with each other, and none are redundant with the direction of E_{i1}.

It is possible, or perhaps likely, that a given line drawing will include the edge E_{OS} between S_O and S_S, as in figure 14. An *interesting question arises as to whether this provides some* additional constraint, which might perhaps relax the requirement that three pieces of information be provided in advance.

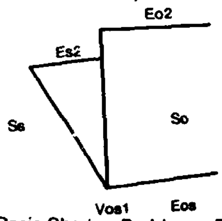


Figure 14: Basic Sh~~·~~low Problem -- Edge E_{OS} Provided

The edge E_{OS} turns out to be redundant with E_{O2} and E_{S2}, in the sense that given the latter, the former can be constructed, and vice versa. Suppose we are given E_{O2} and E_{S2}. These represent the intersections (in the *scene*) of planes S_O and S_{i2}, and S_S and S_{i2}, respectively. Now, either these two lines intersect or they do not. Suppose they intersect in a point. Call it V_{OS2}, since it is contained in surfaces S_O, S_S, and S_{i2}. Th·. point is contained in both S_O and S_S, as is point V_{OS1} which is given in the line drawing. Therefore, the line E_{OS} must pass through these points. On the line drawing, find the intersection of E_{O2} and E_{S2}. Draw the line joining this point to V_{OS1}: this is E_{OS} (Figure 15).

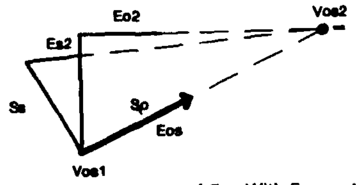


Figure 15: Redundancy of E_{OS} With E_{O2} and E_{S2}

Now, suppose that the two lines E_{O2} and E_{S2} do not intersect anywhere. Then there is no point V_{OS2} contained in all three surfaces S_O, S_S, and S_{I2}. So, E_{OS} cannot intersect either E_{O2} or E_{S2}. Since it is coplanar with these (on surfaces S_O and S_S, respectively), it must be parallel to both. Edge E_{OS} can therefore be drawn through V_{OS1}, parallel to E_{O2} (and E_{S2}).

By this reasoning, E_{OS} can be constructed from E_{O2} and E_{S2}. Similarly, if E_{OS} is given, either of E_{O2} and E_{S2} can be calculated from the other, to provide the geometric constraint described above for the solution of the Basic Shadow Problem. Of course, the solution can also proceed directly using the label - on E_{OS}, with identical results.

The solution of this problem should be compared with the solution to the problem if there are no shadows -- if just S_O is given, joined to S_S along edge E_{OS}. Here, there are four parameters (G_O and G_S) to compute, and one constraint from the image (E_{OS}), so three pieces of information are still needed in advance. With shadows, the same number of *a priori* parameters are needed, but one of them can be a description of the light source position instead of a description of a surface orientation. The significance of shadows is that they allow information about the light source to be used to solve the problem as a substitute for information about the surface orientations themselves.

It has not been assumed in this discussion that surfaces S_O and S_S must touch. In practice, the Basic Shadow Problem arises any time there are two surfaces which provide two shadow edge pairs and an enclosed illumination vector (Figure 16). Any additional shadow edge pairs on these two surfaces will be redundant, as will any visible edges along which these two surfaces intersect directly.

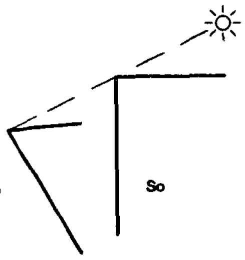


Figure 16: Occurence of the Basic Shadow Problem

2.3 Varying the Location of the Light Source

When the light source is in front of the camera (i.e. in the scene, where it might even appear in the image) and infinitely far away, the Basic Shadow Problem takes the form shown in figure 17. In this case, the first illumination surface S_{I1} joins edges E_{O1} and E_{S1}, giving both of these edges - labels. Illumination surface S_{I2} joins E_{O2} and E_{S2}. At E_{S2}, the label is clearly - To label E_{O2}, it is necessary to extend S_{I2} above this edge, and apply the label to S_O and the upper half-plane of S_{I2}. The label will then be +.

The vector pointing toward the light source does not intersect the plane $z = 1$, but the vector pointing *away* from the light source (toward the camera) does. This has the effect of placing the point G_I in the gradient space on a line parallel to edge E_{I1} passing through the origin as before, but on the half-line towards surface S_S instead of towards surface S_O. Also, while the redundancy of edge E_{OS} is also the same as in the Basic Shadow Problem, edges E_{O3} and E_{S3} are redundant with edges E_{O2} and E_{S2}. This can be easily seen, since edge E_{OS} can be calculated from the intersection of E_{O1} and E_{S1} and the intersection of E_{O3} and E_{S3}; since edge E_{OS} is known to be redundant with E_{O2} and E_{S2}, so must be E_{O3} and E_{S3}.

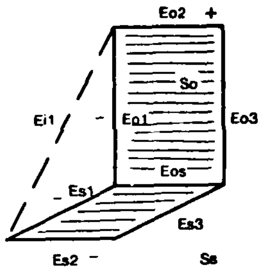


Figure 17: Light Source In Front of Camera, Infinitely Far Away

If the light source is behind the camera but below it, and infinitely far away, then the geometry is as shown in figure 18. In this case, the only difference from the Basic Shadow Problem is that edge E_{O2} receives the label + instead of -; the labels of edges E_{O1}, E_{S1}, E_{S2}, and E_{OS} (if present) will be the same as previously described.

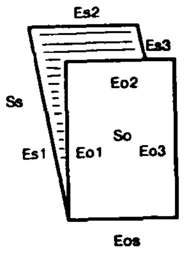


Figure 18: Light Source Behind and Below Camera, Infinitely Far

If the light source is a point not infinitely far away, then all illumination vectors will converge at the light source instead of being parallel (Figure 19). Only two of the preceding arguments need to be changed in this case. The first difference is that the value k is dependent upon the particular illumination vector used, and each illumination vector will have its own value of k and its own line of illumination surface gradients L_{illum}.

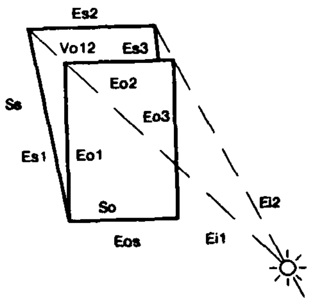


Figure 19: Point Light Source at Finite Distance

The second change is that edges E_{O3} and E_{S3} are no longer interchangeable with E_{OS} or with E_{O2} and E_{S2}. The new information is actually provided not by the angle between the edges E_{O3} and E_{S3}, but by the new illumination vector E_{I2} seen between vertices V_{O23} and V_{S23}. This is shown in figure 19 for one case (light source below and behind camera); similar line labels and reasoning hold for the other cases presented previously.

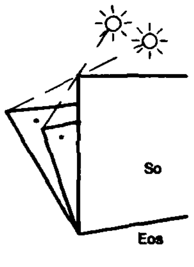
In this arrangement, the exact position of the light source can be calculated. The lines E_{I1} and E_{I2} must intersect (in the scene); the light source is located at the point of intersection. Under orthography, as we are assuming here, the x and y coordinates of the light source will be the same as the x-y coordinates of the intersection of the lines in the image. So, these coordinates can easily be found. The relative z coordinate is then found using the k value for either of these vectors (E_{I1} or E_{I2}), using the definition of k presented above in equation (2.1): if (Δx, Δy, Δz) is an illumination vector from an object vertex to the light source (such as E_{I1} or E_{I2}), then Δx and Δy can be measured in the image, and

$$\Delta z = \mathrm{sqrt}(\Delta x^2 + \Delta y^2) / k$$

It can be determined from the line drawing whether the light source is in fact infinitely far away: if two illumination vectors (such as E_{I1} and E_{I2}) *intersect, then the light source is at a finite distance,* and all illumination vectors in the image must intersect at the same point. If any two illumination vectors are parallel, then all illumination vectors are parallel and the light source is infinitely far away. These observations can be used to arrive at constraints between various simple shadow problems that arise in different parts of the same image, involving different objects and surfaces.

2.4 Changing the Number of Light Sources

It is possible that several light sources *will be present, as in figure* 20. In this case, each light source produces two parameters in the problem (the direction of illumination), and adds two image constraints (an illumination vector and one non-redundant shadow edge pair). The number of *a priori* parameters needed will be the same, regardless of how many light sources are present.



• edges are redundant with Eos

Figure 20: Basic Shadow Problem With Multiple Light Sources

However, for each light source, one of the *a priori* parameters may be the value k for that light source, based on knowledge of the three-dimensional direction of illumination. In general, if n light sources are present and the value of k is known for each, the problem has $2n+4$ parameters, the image provides $3n+1$ constraints, and 3-n parameters are needed in advance. Thus, shadows allow you to use *a priori* knowledge about light source positions instead of *a priori* knowledge about surface orientations when computing the gradients of the visible surfaces.

In figure 21, there are no light sources or shadows. There are 4 parameters to compute (the gradients of the two surfaces). An image constraint will be provided in this case only if the two surfaces S_O and S_S touch along edge E_{OS}; if they do not, then an extra *a priori* parameter will be needed (i.e. 4 instead of 3).

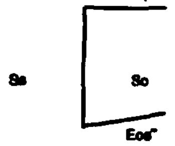


Figure 21: Two Surfaces With No Light Source

3. Polyhedral Shadow Geometry

3.1 Shadows Falling On Polyhedra

The shadow of S_O may fall on two surfaces, S_S and S_T, connected by an edge E_{ST} (Figure 22). In this case, the first illumination surface S_{I1} contains edges E_{O1}, E_{S1}, and E_{T1}. *Illumination surface* S_{I2} *contains edges* E_{O2} and E_{S2}. Edge E_{I1} is an *illumination vector, joining vertices* V_{O12} and V_{S12}.

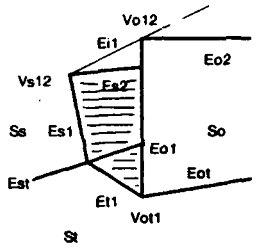


Figure 22: Shadow Falling On Two Surfaces

In this figure, a Basic Shadow Problem can be solved using surfaces S_O and S_S. In addition, there are two new parameters to compute (G_T), and two new constraints (from edges E_{ST} and E_{T1}). These new constraints can be used to compute G_T once the Basic Shadow Problem has been solved, using these relations:

$$E_{ST} \in S_T, S_S \qquad -\Delta z_{ST} = G_T \cdot E_{ST} = G_S \cdot E_{ST}$$
$$E_{T1} \in S_T, S_{I1} \qquad -\Delta z_{T1} = G_T \cdot E_{T1} = G_{I1} \cdot E_{T1}$$

$$\begin{bmatrix} -\Delta z_{ST} \\ -\Delta z_{T1} \end{bmatrix} = \begin{bmatrix} E_{ST}^T \\ E_{T1}^T \end{bmatrix} G_T$$

$$G_T = \begin{bmatrix} E_{ST}^T \\ E_{T1}^T \end{bmatrix}^{-1} \begin{bmatrix} G_S \cdot E_{ST} \\ G_{I1} \cdot E_{T1} \end{bmatrix}$$

The edge E_{ST} is labeled $-$ if the shadow edge E_{S1} bends toward E_{O1} from E_{t1}, and $+$ if it bends away from E_{O1}. The complete solution of this problem, like the Basic Shadow Problem, requires that three pieces of information be supplied in advance.

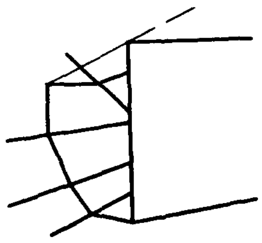


Figure 23: Shadow Falling On Many Surfaces

This solution technique can be generalized to cases such as figure 23, in which there are several shaded surfaces. If there are n shaded surfaces which intersect the shadow edge with no discontinuities in the shadow edge, the problem will have a total of $2n+4$ parameters: $2n$ for the gradients of the shaded surfaces, 2 for G_O, and 2 for G_I. The image will supply $2n+1$ constraints; three parameters must be given in advance.

It is possible for the shadow edge to exhibit discontinuities when the shadow edge falls across occluding edges, as in figure 24.

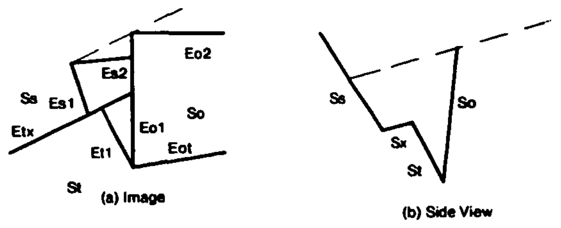


Figure 24: Shadow Edge With Discontinuites

The solution method is exactly as before, but this time there will be no constraint between surfaces S_S and S_T, since edge E_{ST} has been replaced by edge E_{TX} which provides no constraint between S_S and S_T. Therefore, the image provides one less constraint, and one additional non-redundant parameter must be supplied in advance in order to compute all the surface orientations. Of course, the gradient of surface S_X cannot be computed, since S_X is not visible in this image.

It is also the case that the edge E_{OT} (between the occluding surface and one shaded surface) is *non-redundant* if there are any discontinuities along the shadow edge caused by illumination surface S_{I1} (as in figure 24). Therefore, if this edge is present, the image provides an additional constraint.

3.2 Shadows Cast by Simple Polyhedra

When a shadow is cast by a polyhedron as in figure 25, each shadow-making edge (E_{PX}, E_{OP}) must be the intersection of an illuminated surface and a self-shadowed surface of the polyhedron. In the figure, S_O is illuminated and S_P is self-shadowed. The edge E_{OP} between them is a shadow-making edge, and corresponds to shadow edge E_{S1}. Illumination surface S_{I1} contains these two edges. Similarly, it can be concluded that edge E_{PX} is a shadow-making edge, and must correspond to shadow edge E_{S2} (via illumination surface S_{I2}).

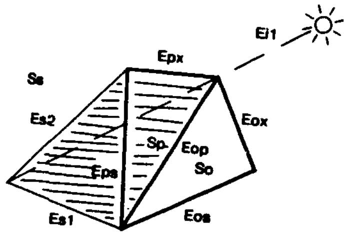


Figure 25: Shadow Cast By Simple Polyhedron

It can be deduced from the above observations that whatever *surface intersects S_P along edge E_{PX} must be illuminated.* It cannot, however, be concluded that the surface containing edge E_{PX} also contains edge E_{OX}. For this reason, no strong statements can be made about the surfaces that are not visible in the image.

In the figure, a Basic Shadow Problem exists involving surfaces S_P and S_S. The edge E_{PS} is therefore redundant with the two shadow edge pairs (E_{OP} and E_{S1}, E_{PX} and E_{S2}). This is important, since it is typically difficult to resolve details such as edge E_{PS} within shaded portions of the image [7].

When the basic problem has been solved, the gradients of surfaces S_P and S_S will be known. The gradient of S_O can then be calculated by using the constraints provided by edges E_{OP} (with surface S_P) and E_{OS} (with surface S_S).

Little useful information is provided by edge E_{OX}, since it borders on only one visible or constructible surface (S_O). Edge E_{PX}, on the other hand, is very important, since it borders on two surfaces (visible surface S_P and the illumination surface S_{I2}).

In this problem, there are eight parameters to be computed (the gradients of surfaces S_O, S_P, and S_S, and the direction of the light source G_I). The image provides five constraints (two from the shadow edge pairs E_{OP}-E_{S1} and E_{PX}-E_{S2}, one from the illumination edge E_{I1}, and two from the edges E_{OP} and E_{OS}). Therefore, three parameters must be provided in advance in order to perform the computation.

If the figure were drawn with no shadows, there would be six parameters altogether (the gradients of the three surfaces), and three constraints in the image (from edges E_{OP}, E_{OS}, and E_{PS}). Three parameters would be required in this case, also. As in the Basic Shadow Problem itself, the shadow of a polyhedron does not provide additional constraints; it merely allows you to substitute information about the light source for *a priori* information about the surface orientations themselves, and allows you to utilize easy-to-find shadow edges instead of hard-to-find details within shaded areas of the image.

The above method of solution also applies when the light source is in a different position as in figure 26, which illustrates two illuminated surfaces of a polyhedron.

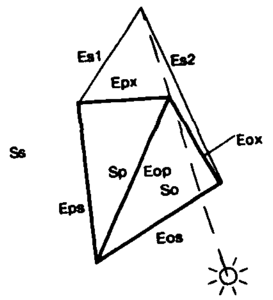


Figure 26: Light Source In a Different Position

3.3 Shadows Cast by Complex Polyhedra

Suppose we add an additional self-shadowed surface to figure 25, as in figure 27. In this figure, both S_A and S_P are self-shadowed. We will suppose that the new surface S_A adjoins a shadow-making edge E_{AO}. (If the new surface S_A does not adjoin a shadow-making edge, it will be buried in the middle of the shaded area and will have no effect on the shape of the shadow.)

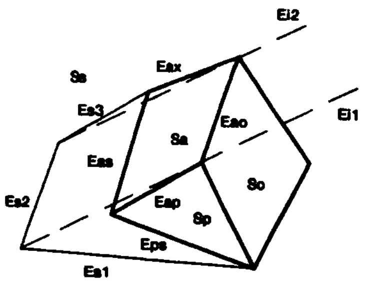


Figure 27: Polyhedron With Two Self-Shadowed Surfaces

Two new parameters are present in the system: the gradient G_A of the new surface S_A. The image provides two new constraints that can be used to solve for these two parameters: the shadow edge pair E_{AX}-E_{S3}, and the edge E_{AO} between surfaces S_A and S_O. So, three parameters are still required in advance to solve the system completely.

The edge E_{AS} is redundant with the shadow edge pair E_{AO}-E_{S2} when shadows are present. One of the two edges E_{AP} and E_{PS} is needed, along with E_{OP}, to determine the gradient of surface S_P. Thus, two of the edges E_{SP}, E_{AS}, and E_{PS} are redundant, and only one is needed. Since these edges all lie in the shadowed area of the image, they will be difficult to extract reliably [11]. Shadows reduce the need to find edges within shadowed areas of the image.

When the basic figure (Figure 25) is modified by adding an *illuminated* surface instead of a *self-shadowed* surface, a line drawing such as figure 28 is the result. In this figure, surfaces S_A and S_O are illuminated, while S_P is self-shadowed. (Again, if the surface does not adjoin a shadow-making edge, there will be no effect on the shape of the shadow and the consequent inferences to be made from shadow geometry. Therefore, we will assume that the new surface S_A does adjoin a shadow-making edge E_{AP}.)

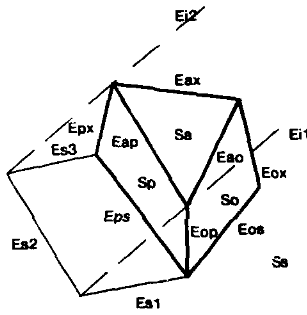


Figure 28: Polyhedron With Two Illuminated Surfaces

The reasoning here is analogous to the case of an additional self-shadowed surface: two new parameters are needed (G_A), and there are two new constraints with shadows (the pair E_{PX}-E_{S3} and the edge E_{AO}), and two new constraints with no shadows (edges E_{AO} and E_{AP}). In any case, three parameters will be required in advance.

It is possible that additional *a priori* parameters will be needed in pathological cases. Figure 29 depicts an object with a surface adjoining the shadow-making edge which is not visible in the image (at E_{PX}). Here, an additional *a priori* parameter will be needed to determine the gradient of surface S_R. The additional parameter is needed because edge E_{QX} provides no constraint between surfaces S_Q and S_R. This situation is analogous to the discontinuities in the shadow edge discussed previously.

Another circumstance requiring additional *a priori* parameters is shown in figure 30. Here, vertex V_{OPQR} is not *trihedral* -- there are four surfaces meeting at that point (S_O, S_P, S_Q, and S_R). This adds one degree of uncertainty involving the gradients of surfaces S_Q and S_R: one additional *a priori* parameter is needed to solve this problem.

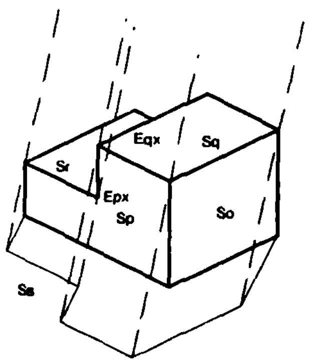


Figure 29: Additional Parameter Needed for Hidden Surface

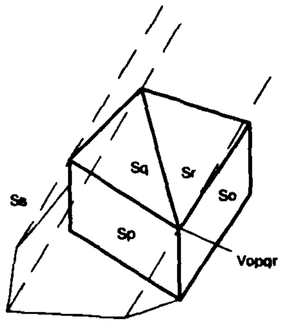


Figure 30: Additional Parameter Needed for Non-Trihedral Vertex

3.4 The General Solution For Polyhedral Shadow Geometry

The results of the two previous extensions can be directly combined. In these arguments, it has never been assumed that the shadow edge E_{S3} and the corresponding shadow-making edge (E_{AX} or E_{PX}) meet at a vertex. Therefore, the results apply without change to line drawings with additional hidden surfaces, such as figure 31. In this figure, there is no strong information to be obtained from shadow edge E_{S4}.

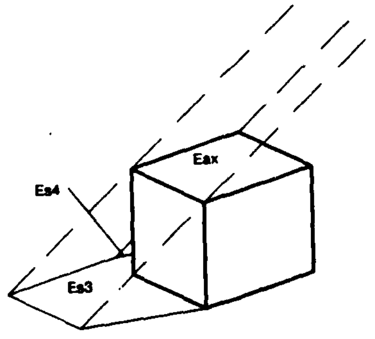


Figure 31: Polyhedron With Additional Invisible Surfaces

Suppose the image depicts i illuminated surfaces and s self-shadowed surfaces along the shadow-making edges of a polyhedron, casting a shadow whose corresponding edge intersects n surfaces of another polyhedron exhibiting d discontinuities, with h hidden shadow-making surfaces and t non-trihedral vertices.

- The problem has $2i + 2s + 2n + 2$ parameters:
  - o $2i$ for the gradients of the i illuminated surfaces
  - o $2s$ for the gradients of the s self-shadowed surfaces
  - o $2n$ for the gradients of the n background surfaces
  - o 2 for the direction of illumination, G_I
- The image provides $2i + 2s + 2n\text{-}d\text{-}h\text{-}t\text{-}1$ constraints:
  - o 1 from the illumination vector
  - o 2 shadow-making/shadow edge pairs used to solve the Basic Shadow Problem at one vertex
  - o $i + s\text{-}2$ additional shadow-making edges
  - o $n\text{-}1$ additional shadow edges
  - o $i + s\text{-}h\text{-}t\text{-}1$ non-redundant edges between visible surfaces of the polyhedron casting the shadow
  - o 1 non-redundant edge between the shadow-making polyhedron and the shaded polyhedron
  - o $n\text{-}d\text{-}1$ edges at intersections of visible shaded surfaces
- Therefore, $3 + d + h + t$ parameters must be provided *a priori*:
  - o 3 for the solution of the Basic Shadow Problem
  - o d to compensate for the d discontinuities in the shadow edge due to invisible shaded surfaces
  - o h to compensate for the h hidden shadow-making surfaces
  - o t to compensate for the t non-trihedral vertices

Without shadows, the problem contains $2i + 2s + 2n$ parameters, the image supplies $2i + 2s + n\text{-}d\text{-}h\text{-}t\text{-}2$ parameters, and $n + d + h + t + 2$ parameters must be supplied before the computation.

If $i > 1$ or $s > 1$, an additional illumination vector can be used to determine the exact position of the light source.

The contribution of shadows for computing surface orientations from polyhedral line drawings can be summarized:

- Shadows provide an increasing amount of information when the shadow edge intersects many visible, differently oriented surfaces of the background.
- Shadows allow you to substitute one parameter describing the direction of illumination to replace one parameter describing a surface orientation before performing the required calculations.
- Shadows allow you to substitute (usually) highly visible shadow edges and shadow-making edges for many of the unreliable edges within shaded portions of the image, while providing the same amount of information.

In addition, when several shadow problems appear in different portions of the same image, they share some constraints. For example, suppose several polyhedral blocks are scattered over a single surface. If the gradient of the surface and the direction of illumination are known, then three constraints are provided for *each* of the shadow problems. This will allow the exact solutions to be found for all the problems, if no shadow edge discontinuities or non-trihedral vertices are present.

4. Shadows Involving Curved Surfaces

In this chapter, the involvement of curved surfaces in shadow geometry will be explored. Whether the curvature lies in the occluding surface (object) or the shaded surface, additional information is required to determine the exact surface orientation along the shadow-making arc or the shadow edge arc.

Witkin [16] has also used shadows to determine curved surface orientation. He developed a relation between the curvature of a shadow edge in the scene and the curvature of the shadow edge in the image, then derived surface orientations, using surface texture gradients to provide the additional constraint necessary. The discussion below differs from Witkin's in that surface *orientation* rather than *curvature* (rate of change of orientation) is the basis of the theory.

For discussing curved surfaces, it is necessary to generalize the relation between line labels and surface gradients. Suppose two (possibly curved) surfaces S_A and S_B intersect along arc E_{AB} (Figure 32).

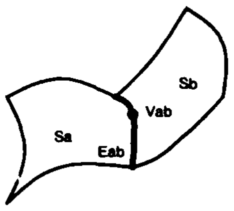


Figure 32: Curved Surfaces Intersecting Along an Arc

The surfaces are defined by

$$S_A\colon -z = f_A(x,y) \qquad S_B\colon -z = f_B(x,y)$$

At a point V_{AB} on E_{AB},

$$-z = f_A(x,y) = f_B(x,y)$$

Differentiating by x and multiplying by Δx,

$$-\Delta x \frac{dz}{dx} = \Delta x\, G_A \cdot (1, \frac{dy}{dx}) = \Delta x\, G_B \cdot (1, \frac{dy}{dx})$$

where G_A and G_B are the gradients of S_A and S_B at V_{AB}, and E = $(\Delta x, \Delta y)$ is a vector tangent to E_{AB} at V_{AB} in the image, corresponding to the three-dimensional vector $(\Delta x, \Delta y, \Delta z)$ in the scene.

Since

$$\Delta z = \Delta x \frac{dz}{dx} \text{ and } \Delta y = \Delta x \frac{dy}{dx},$$

we have

$$-\Delta z = G_A \cdot (\Delta x, \Delta y) = G_B \cdot (\Delta x, \Delta y)$$
$$= G_A \cdot E = G_B \cdot E$$

This is the curved-surface analogue of the relation $-\Delta z = G \cdot E$ described earlier for planar surfaces: the planar-surface edge E is replaced by the tangent vector E to the arc of intersection of two curved surfaces. As a consequence, G_A and G_B lie along a line in gradient space perpendicular to the tangent to the arc of intersection in the image.

4.1 Curvature in the Shaded Surface

Suppose a flat surface is casting a shadow on a curved surface, as in figure 33. Here, vertex V_{S12} is the shadow of vertex V_{O12}. Surface S_{I1}, the first illumination surface, casts the shadow of edge E_{O1} on arc E_{S1} of the curved surface S_S. Surface S_{I2} similarly casts the shadow of edge E_{O2} on arc E_{S2}.

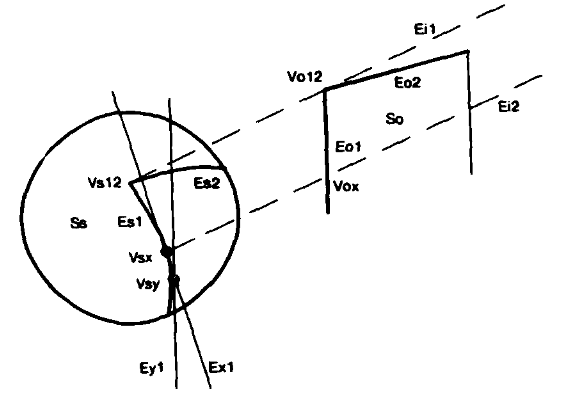


Figure 33: Shadow Cast On a Curved Surface

Suppose V_{SX} is an arbitrary point on the arc E_{S1}. Can we determine the gradient G_X of S_S at this point?

Arc E_{S1} is the arc of intersection between the curved surface S_S and the illumination surface S_{I1} (defined by edge E_{O1} of surface S_O). Therefore, as previously explained, gradients G_X (of S_S at V_{SX}) and G_{I1} (of S_{I1}) must lie along a line in gradient space perpendicular to the tangent line E_{X1} to E_{S1} at V_{SX}. This constraint is illustrated in figure 34.

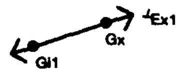


Figure 34: Gradient Space Constraint Between G_X and G_{I1}

This reasoning can be used to find the two tangent lines at vertex V_{S12}, and use them in a Basic Shadow Problem with edges E_{O1} and E_{O2} of the occluding surface S_O. If S_V is the plane tangent to S_S at V_{S12}, the Basic Shadow Problem actually involves surfaces S_V and S_O. For this computation, three *a priori* parameters will be required, and the gradients G_O, G_V, G_{I1}, G_{I2}, and G_I will be computed.

It is not possible to compute the gradients G_X (and G_Y at other vertices V_{SY} as in figure 33, etc.) without additional information. However, it is possible to establish a one-dimensional constraint on each such gradient. Since the gradient G_{I1} of illumination surface S_{I1} was computed as part of the Basic Shadow Problem at vertex V_{S12}, the constraints provided by the tangent lines E_{X1} and E_{Y1} cause gradient space constraints as shown in figure 35. Similar reasoning allows constraints on the gradients at points along arc E_{S2} to be computed, using the gradient G_{I2} of illumination surface S_{I2}.

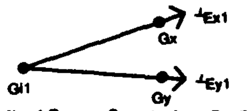


Figure 35: Gradient Space Constraints On Tangent Planes To S_S

For an investment of three parameters given in advance, then, the gradients of S_O and S_V can be computed, as well as a one-dimensional constraint on the gradient for each point along arcs E_{S1} and E_{S2}. Additional constraint for the gradients along these arcs might come from another source such as Horn's "shape from shading" technique [3] or *a priori* knowledge of the shape of the object bounded by surface S_S.

In this shadow problem, if another illumination vector is available (possibly from the shadow of another vertex of S_O), the exact position of the light source can then be determined.

The information available from using shadows in this problem is not redundant with information available from the same line drawing without shadows.

4.2 Shadows Cast By Curved Surfaces

When a curved object casts a shadow on a flat surface as in figure 36, the shadow edge E_{IS} corresponds to the shadow of the "arc of extinction" E_{IO} which divides surface S_O into an illuminated part and a self-shadowed part. There exists a curved illumination surface S_I, composed of illumination vectors, tangent to S_O along E_{IO} and intersecting the shaded surface S_S along E_{IS}. S_I is a cylinder, whose axis is parallel to the direction of illumination.

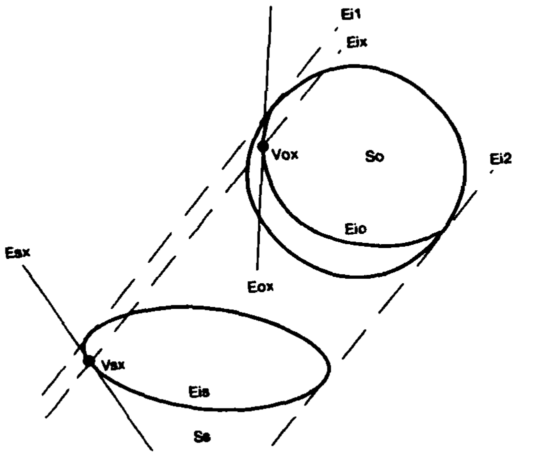


Figure 36: Shadow Cast By a Curved Surface

There is a special significance to the line in the image tangent to both E_{IS} and the outline of S_O: it is an illumination vector, such as E_{I1} in figure 36. If two such tangent lines are visible (as with E_{I1} and E_{I2} in figure 36) or some other feature is visible in both E_{IO} and E_{IS}, then a second illumination vector can be found. From two illumination vectors, the exact position of the light source can be computed and the shadow point V_{SX} can be determined for each point V_{OX} on arc E_{IO}.

The surface S_I is composed entirely of illumination vectors; its gradient at each point must therefore lie along the line L_{illum} in gradient space. To determine this line, the value k for the light source position must be given.

If the light source is not infinitely far away, each illumination vector such as E_{IX}, has a different value of k and determines a different line L_{illum} in gradient space. However, all the values of k can be computed from the position of the light source, given a single value of k such as that for E_{I1}. We will therefore assume, for simplicity, that the light source is infinitely far away, and that a single line L_{illum} exists.

Unfortunately, no stronger statements can be made about the gradient of S_O from examination of the arc E_{IO}. In particular, the direction of the tangent line E_{OX} bears no relationship to the gradient of S_O. This is illustrated in figure 37, which depicts two cylinders tangent to the same illumination plane. The arcs of extinction (dotted lines) have completely unrelated directions in the image.

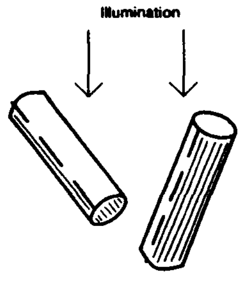


Figure 37: Arcs of Extinction Unrelated To Surface Orientation

However, it is possible to use the shadow edge E_{IS} to compute the gradients of the tangent surfaces along E_{IO}. The gradient G_X of S_O at V_{OX} is the same as the gradient of S_I at V_{OX}, since S_I is tangent to S_O at that point. We have two constraints on G_X from properties of S_I:

1. S_I is an illumination surface, so G_X lies on L_{illum}.
2. The gradient (G_X) of S_I at V_{OX} is the same as the gradient of S_I at V_{SX} (the shadow of V_{OX}), since S_I is a cylinder. As previously shown, G_X and G_S (the gradient of the shaded surface S_S) must lie along a line in the gradient space which is perpendicular to E_{SX}, the line tangent to E_{IS} at V_{SX}.

The constraints on G_X are illustrated in figure 38.

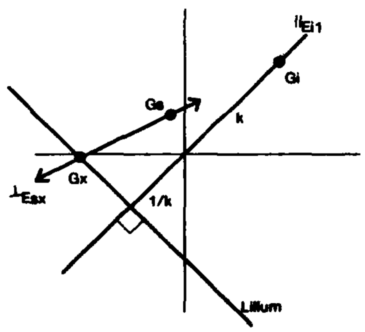


Figure 38: Gradient Space Constraints on G_X

Suppose we are given three parameters -- k and the gradient G_S of surface S_S. From these, it is possible to compute the gradient G_X of the tangent plane to S_O for each point V_{OX} along the arc of extinction E_{IO}. Using the definition of k,

$$\Delta z_{IX} = \|E_{IX}\| / k$$

Since E_{IX} is contained in S_I,

$$-\Delta z_{IX} = G_X \cdot E_{IX}$$

Also, if E_{SX} is a vector tangent to E_{IS} at V_{SX},

$$-\Delta z_{SX} = G_X \cdot E_{SX} = G_S \cdot E_{SX}$$

Combining these,

$$\begin{bmatrix} -\Delta z_{IX} \\ -\Delta z_{SX} \end{bmatrix} = \begin{bmatrix} E_{IX}{}^T \\ E_{SX}{}^T \end{bmatrix} G_X$$

$$G_X = \begin{bmatrix} E_{IX}{}^T \\ E_{SX}{}^T \end{bmatrix}^{-1} \begin{bmatrix} -\Delta z_{IX} \\ -\Delta z_{SX} \end{bmatrix}$$

$$= \begin{bmatrix} E_{IX}{}^T \\ E_{SX}{}^T \end{bmatrix}^{-1} \begin{bmatrix} -\|E_{IX}\|/k \\ G_S \cdot E_{SX} \end{bmatrix}$$

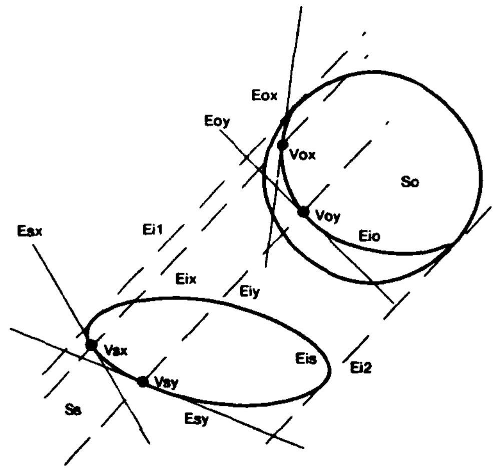


Figure 39: Using E_{IO} to Calculate the Gradient of S_S

It is also possible to use knowledge about the shape of the curved object S_O when G_S is not known in advance. Suppose that two vectors E_{OX} and E_{OY} tangent to the arc of extinction E_{IO} at points V_{OX} and V_{OY} are known. Let points V_{SX} and V_{SY} be the shadows of V_{OX} and V_{OY}, let E_{IX} and E_{IY} be the illumination vectors joining V_{OX} to V_{SX} and V_{OY} to V_{SY}, and let E_{SX} and E_{SY} be vectors tangent to the shadow edge E_{IS} at V_{SX} and V_{SY} (Figure 39).

If (Δx_{OX}, Δy_{OX}, Δz_{OX}) is the three-dimensional vector corresponding to E_{OX}, with similar definitions for the other vectors, then Δz_{OX} and Δz_{OY} are known in advance. As previously shown, if G_X is the gradient of S_I (and S_O) at V_{OX}, then

$$-\Delta z_{OX} = G_X \cdot E_{OX}$$

Since E_{IX} is an illumination vector,

$$\Delta z_{IX} = \|E_{IX}\| / k$$

and, since E_{IX} is contained in S_I at V_{OX},.

$$-\Delta z_{IX} = G_X \cdot E_{IX}$$

Combining,

$$\begin{bmatrix} -\Delta z_{OX} \\ -\Delta z_{IX} \end{bmatrix} = \begin{bmatrix} E_{OX}{}^T \\ E_{IX}{}^T \end{bmatrix} G_X$$

$$G_X = \begin{bmatrix} E_{OX}{}^T \\ E_{IX}{}^T \end{bmatrix}^{-1} \begin{bmatrix} -\Delta z_{OX} \\ -\|E_{IX}\|/k \end{bmatrix}$$

So, G_X (and, similarly, G_Y, the gradient of S_I at V_{OY}), can be determined exactly.

Now, since S_I and S_S intersect at V_{SX} along E_{SX}.

$$-\Delta z_{SX} = G_X \cdot E_{SX} = G_S \cdot E_{SX}$$

Similarly,

$$-\Delta z_{SY} = G_Y \cdot E_{SY} = G_S \cdot E_{SY}$$

So,

$$\begin{bmatrix} -\Delta z_{SX} \\ -\Delta z_{SY} \end{bmatrix} = \begin{bmatrix} E_{SX}^T \\ E_{SY}^T \end{bmatrix} G_S$$

$$G_S = \begin{bmatrix} E_{SX}^T \\ E_{SY}^T \end{bmatrix}^{-1} \begin{bmatrix} -\Delta z_{SX} \\ -\Delta z_{SY} \end{bmatrix} = \begin{bmatrix} E_{SX}^T \\ E_{SY}^T \end{bmatrix}^{-1} \begin{bmatrix} G_X \cdot E_{SX} \\ G_Y \cdot E_{SY} \end{bmatrix}$$

and therefore, G_S can be determined exactly. Now, G_S can be used as previously shown to determine the gradient of S_O at each point *on the arc of extinction E_{IO}. Here, knowledge of k and the direction* tangent to E_{IO} at two points has sufficed to determine the gradient of S_S and the gradient of S_O at all points along E_{IO}.

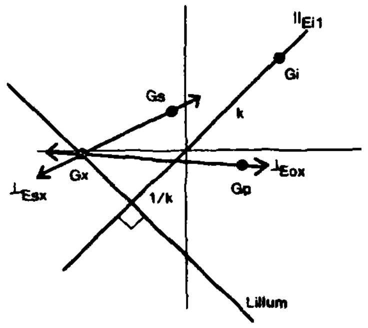


Figure 40: Gradient Space Constraints From V_{OX} On G_S

In the special case that S_O is spherical, for example, the entire arc of extinction E_{IO} lies in a plane S_P whose suriace normal is an *illumination vector. Therefore, the gradient $G_P = G_I$. In this case,* the entire problem can be solved with only one parameter (k) given in advance, since Δz_{OX} and Δz_{OY} can be calclated directly:

$$G_I = E_{I1} \frac{k}{\|E_{I1}\|}$$

$$-\Delta z_{OX} = E_{OX} \cdot G_I = \frac{k\,(E_{CX} \cdot E_{I1})}{\|E_{I1}\|}$$

$$\text{and} \quad -\Delta z_{OY} = \frac{k\,(E_{OY} \cdot E_{I1})}{\|E_{I1}\|}$$

The shadow information just described is not redundant with information available in the same line drawings when no shadows are present.

5. Shadow Geometry and Other Shape Inference Techniques

Shadow geometry can be combined with other techniques for determining 3D interpretations from images.

5.1 Other Gradient Space Techniques

In [13], the closed-form solution for the Basic Shadow Problem is presented in the form:

$$G_O = f(G_S, k)$$

When k is given in advance, G_O is shown to be an affine transform (two-dimensional linear transform) of G_S.

Stated in this form, it is very convenient to use shadow geometry in conjunction with other techniques for determining surface gradients. For example, in figure 41, a line drawing is shown in which the intensities of the surfaces are known. If the surfaces are Lambertian or have known reflectance functions, Horn's "shape from shading" technique [3] can be used to determine a contour in gradient space along which G_S must lie, and a similar contour for G_O. Now, if the contour for G_S is transformed in its entirety by the function f provided by shadow geometry (as discussed above), a new contour for G_O is provided in gradient space (figure 42). Since G_O must lie along two contours, it must lie at one of the points of intersection of these contours. Now, for each such point, the corresponding point G_S can be determined using the inverse of transform f.

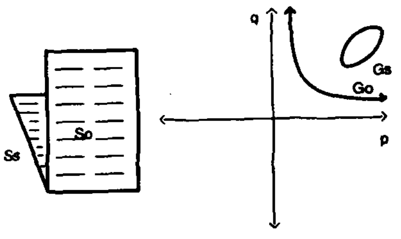


Figure 41: Shape From Shading

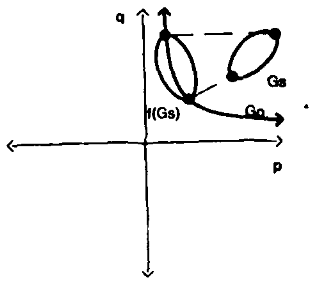


Figure 42: Shadow Geometry and Shape From Shading

Shadow geometry can similarly be combined with Kanade and Kender's "skewed symmetry" [6], as in figure 43. Here, *skewed symmetry provides a hyperbolic contour for each of the two surface* gradients G_O and G_S; shadow geometry can be used to transform the contour for G_S into an additional contour for G_O. The points of intersection of the contours for G_O are then the possible values of G_O, and the corresponding values of G_S can be found as above.

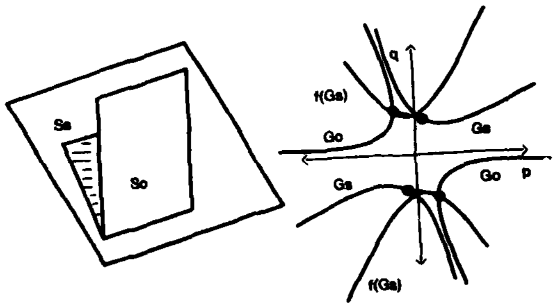


Figure 43: Shadow Geometry and Skewed Symmetry

5.2 Shape Recovery for Curved Surfaces

Some techniques have appeared in the literature for reconstructing the orientation of a curved surface at every point, using relaxation techniques [1, 5]. These techniques typically begin with the surface orientation at every point along the outline of the surface (S_O in figure 44). These values form a boundary condition which drives the relaxation process.

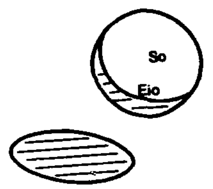


Figure 44: Shadow Geometry and Curved Surface Recovery

In this paper, we have seen that it is possible to determine the surface orientation for the tangent planes at each point along the arc of extinction E_{IO}, using three *a priori* parameters (such as the k value for the light source and the orientation of the surface on which the shadow appears). These values can be used to provide stronger boundary conditions for relaxation techniques.

Surface orientations along the arc of extinction are valuable for another reason. Relaxation techniques must make some presumptions about the curvature of the surface (e.g. surface of minimum curvature, cubic or other surface of revolution). Since all of these models of curvature are consistent with the tangent gradients along the outline of S_O, it is not possible to decide which model is appropriate when the only boundary condition comes from the outline of S_O. However, when the arc of extinction is also used, it may be possible to select one from several possible models of surface curvature, or to measure systematic deviation from a particular model for a specific object.

6. Conclusions

This paper has presented a theory describing relationships among surface orientations in line drawings with shadows. The relationships arise from hypothesizing the existence of "illumination surfaces" connecting shadow edge pairs, assigning appropriate line labels to shadow and shadow-making edges, and applying the resulting constraints in the gradient space.

This technique falls short of providing exact solutions to shadow geometry problems. The line drawing must be augmented with information such as the orientations or curvature of specific surfaces or the position of the light source if exact surface orientations are to be found.

It has been shown, however, that shadow geometry provides important benefits for image understanding:

- Shadows allow you to subsitute information about the light source position instead of *a priori* knowledge about surface orientations.
- Shadows allow you to determine geometric information from highly visible shadow edge pairs instead of using many of the unreliable edges within shaded portions of an image.
- An increasing amount of information is provided by the shadow edge when the shadow falls on many visible, differently oriented surfaces.
- Shadows provide some constraint when curved surfaces are involved.
- Shadows provide constraint between surfaces even when they do not touch in the scene (or image).
- Shadows allow the solution to one shadow problem to be used in the solution of other shadow problems, since typical shadow problems are mutually constrained (e.g. same light source, same background surface).

In addition, some observations have been made about the solution of the correspondence problem for shadows, which must be solved before surface orientations can be inferred.

7. Bibliography

1. Barrow, H. G. and Tenenbaum, J. M. Reconstructing Smooth Surfaces From Partial, Noisy Information. ARPA IUS Workshop, November, 1979, pp. 76-86.
2. Clowes, M. B. "On Seeing Things." *Artificial Intelligence 2* (1971), 79-116.
3. Horn, B. K. P. "Understanding Image Intensities." *Artificial Intelligence 8* (1977), 201-231.
4. Huffman, D. A. Impossible Objects as Nonsense Sentences. In *Machine Intelligence 6*, Meltzer, B. and Michie, D., Ed.,American Elsevier Pub. Co., New York, 1971, ch. 19, pp. 295-323.
5. Ikeuchi, K. Numerical Shape From Shading and Occluding Contours in a Single View. AIM 566, MIT, November, 1979.
6. Kanade, T. and Kender, J. Mapping Image Properties into Shape Constraints: Skewed Symmetry, Affine-Transformable Patterns, and the Shape-from-Texture Paradigm. IEEE Workshop on Picture Data Description and Management, Asilomar, CA, August, 1980, pp. 130-135.
7. Lowe, D. G. and Binford, T. O. The Interpretation of Geometric Structure from Image Boundaries. ARPA IUS Workshop, April, 1981, pp. 39-46.
8. Mackworth, A. K. "Interpreting Pictures of Polyhedral Scenes." *Artificial Intelligence 4* (1973), 121-137.
9. McKeown, D. M. and Kanade, T. Database Support for Automated Photo-Interpretation. ARPA IUS Workshop, April, 1981, pp. 7-13.
10. Nagao, M., Matsuyama, T., and Ikeda, Y. "Region Extraction and Shape Analysis in Aerial Photographs." *Computer Graphics and Image Processing 10* (1979), 195-223.
11. O'Gorman, F. Light Lines and Shadows. School of Social Sciences, U. of Sussex, December, 1975.
12. Ohlander, R. B. *Analysis of Natural Scenes*. Ph.D. Th., Carnegie-Mellon University, April 1975.
13. Shafer, S. A. and Kanade, T. Using Shadows in Finding Surface Orientations. Carnegie-Mellon University Computer Science Department, January, 1982.
14. Smith, H. T. U.. *Aerial Photographs and Their Applications*. D. Appleton-Century Co., New York, 1943.
15. Waltz, D. Understanding Line Drawings of Scenes with Shadows. In *The Psychology of Computer Vision*, Winston, P. H., Ed.,McGraw-Hill, 1975, ch. 2, pp. 19-91.
16. Witkin, A. Shape From Contour. AI-TR 589, MIT, November, 1980.
17. Witkin, A. P. and Fischler, M. A. Recovering Intrinsic Scene Characteristics from Images. Interim Tech Report Sept. 1980 - Sept. 1981, SRI International, AI Center, November, 1981.

AD-P000 116

MATCHING LINEAR FEATURES OF IMAGES AND MAPS

Gerard G. Medioni
Intelligent Systems Group
Department of Computer Science
University of Southern California
Los Angeles, California 90089-0272

ABSTRACT

We present two methods to solve the problem of matching linear features extracted from an aerial image with a set of linear features derived from a map or another view of the same scene. The first method, using discrete relaxation, is very efficient but requires the model to have a small number of elements. The other one, called the kernel method, demonstrates how drastically we can improve the running time if we know a few pairs of matched elements. Illustrative examples are provided, and extensions are discussed.

1. INTRODUCTION

Suppose that we are given a very high resolution aerial picture taken from a known altitude and with a known orientation, together with a detailed map of the area. How can we determine which parts of the picture correspond to given elements of the map? The complexity of this problem stems partly from the fact that a picture is described in terms of pixel intensities while the map is a set of high level abstract entities. There have been several tentative answers to this question. Early systems worked directly with the intensity array [1,2,3], trying to find transformations which map one array into another. Problems arise when the illumination changes substantially or when the texture changes with the seasons.

Price and Faugeras [4,5], extracted linear features and regions to be matched with a map whose characteristics are derived manually. They use relative position constraints and stochastic labeling.

The Hughes Research Laboratories [6,7,8] conducted studies to match two views of a scene using line and vertex features derived from the scene.

We first extract features from the intensity images using the USC linear feature extraction system [9]. The technique consists of convolving the image with 6 directional edge masks, each 5*5 pixels, choosing the maximum, thinning and thresholding the convolved output, linking the resulting edges based on proximity and orientation, and finally approximating by straight lines.

The two linear features we consider are SEGMENTS and APARS. Segments are linear approximations of a set of connected edge points. They are defined by their endpoints, orientation, length and average contrast. Apars are a *representation of two segments separated by a small* gap and with orientations differing by 180°. Apars are very appropriate to represent roads and rivers. If the scene is to be matched with a map, we encode the linear pieces of the map manually. The problem can now be formulated as: Which elements in the image correspond to the given elements in the map, based on geometrical constraints.

The next section provides assumptions and definitions, the third section describes the relaxation method, the fourth describes the derived kernel method, the fifth presents results and the conclusion outlines possible extensions.

2. ASSUMPTIONS AND DEFINITIONS

We assume that

1. The model and the scene have approximately the same orientation, since the orientation of the plane at the time of the picture is known.

2. The scaling factor from the model to the scene, μ, is known, since the camera characteristics and the altitude of the plane are known.

Let us define the following terms: We will denote the linear features of one image as a_i, $1 \leq i \leq n$, and call them <u>objects</u>.
We will denote the linear features of the other image, or of the map, as λ_j, $1 \leq j \leq m$, and call them <u>labels</u>.
The set $A = \{a_i \mid 1 \leq i \leq n\}$ is the <u>scene</u>.
The set $L = \{\lambda_j \mid 1 \leq j \leq m\}$ is the <u>model</u>.

We are interested in computing the quantity $p(i,j)$ which is the <u>possibility</u> for object a to have label λ_j. $p(i,j)$ is a discrete variable that can be either 0 or 1; it is <u>NOT</u> a probability because

1. object a_i may have no label ($\sum_j p(i,j) = 0$).

2. several objects may have the same label ($\sum_i p(i,j) > 1$).

3. object a_i may have several labels ($\sum_j p(i,j) > 1$).

$p^0(i,j)$ represents the initial assignment.
$p^t(i,j)$ represents the assignment at the t^{th} iteration.

The methods presented here principally rely on geometrical constraints, meaning that when we assign a label λ_j to an object a_i, we expect to find an object a_h with a label λ_k in a certain position depending on i,j,k. This area is denoted w(i,j,k) and is called the <u>window</u>(i,j,k).

Let us represent any object a_i by a vector $\overrightarrow{A_iB_i}$, and any label λ_j by a vector $\overrightarrow{P_jQ_j}$. We know where any other label $\lambda_k(\overrightarrow{P_kQ_k})$ is in the model, relative to λ_j : it is uniquely defined by $\overrightarrow{P_jP_k}$ for example. Having this information, it should be simple to find the corresponding zone in the scene, as illustrated in the example of figure 1.

Let us consider object a_i ($\overrightarrow{A_iB_i}$) with assigned label λ_j ($\overrightarrow{P_jQ_j}$). To define w(i,j,k), we "slide" $\overrightarrow{A_iB_i}$ over $\overrightarrow{P_jQ_j}$ to obtain two extreme positions:

1. Identifying A_i with P_j we define the 2 points R_1,S_1 by

   a) $\overrightarrow{OR}_1 = \overrightarrow{OA}_i + \mu * \overrightarrow{P_jP_k}$.
   b) $\overrightarrow{OS}_1 = \overrightarrow{OR}_1 + \mu * \overrightarrow{P_kQ_k}$.

2. Identifying B_i with Q_j we define the 2 points R_2,S_2 by

   a) $\overrightarrow{OR}_2 = \overrightarrow{OB}_i + \quad * \overrightarrow{Q_jP_k}$.
   b) $\overrightarrow{OS}_2 = \overrightarrow{OR}_2 + \quad * \overrightarrow{P_kQ_k}$.

These 4 points R_1, S_1, R_2, S_2 are the 4 corners of a window w(i,j,k) in which we should find an object a with the label .

Finally, we need to define the relation $\in$, "is compatible with", between (i,j) and (h,k) as
(i,j) IS COMPATIBLE WITH (h,k)
<==> (i,j) $\in$ (h,k)
<==> a in w(i,j,k) AND a in w(h,k,j).

We need to check both predicates because the relation "is in w" is not symmetric, that is a_h in w(i,j,k) does not imply a_i in w(h,k,j).

We now can proceed to explain how the methods operate.

3. DESCRIPTION OF THE RELAXATION METHOD

This method takes a scene and a model as input. We first assign possibilities based on comparable angular orientation of an object a and a label ; then, for each such pair (i,j), we verify that, for any other label , we find an object a with this label in the window w(i,j,k) where we expected to find one. Some tolerance is accepted to take into account inaccuracy and partial matches. If the evidence is sufficient, we keep the pair (i,j) otherwise we discard it. We iterate until we reach a stable configuration, that is until we reach an iteration where no pair is discarded. Figure 2 shows a flowchart of the procedure.

Let us now explain this process in formal terms.

Given a set of objects $A=\{\ a_i |\ 1 \leq i \leq n\}$ (scene) and a set of labels $L=\{\lambda_j\ |\ 1 \leq j \leq m\}$ (model), we are looking for a subset $M=\{\ (a_i,\lambda_j)\ |\ p(i,j)=1\ \}$ of the cartesian product A*L which is the set of objects matched with a label.

Let M^t be the superset of M at the tth iteration. $M^t=\{\ (a_i,\ \lambda_j\ |\ p^t(i,j)=1\ \}$

1. Initial assignment of possibilities
Let θ_i be the angular orientation of a_i, let ϕ_j be the angular orientation of λ_j. Then, for all (i,j), p (i,j)=1 iff $|\theta_i - \phi_j\ | \leq 15$.

2. Iteration formula
For all (i,j) we define $p^{t+1}(i,j)$ = 1 iff $p^t(i,j)$=1 and $\forall$ k in [1,m],$\exists$h in [1,n] such that (i,j) $\in$ (h,k), that is a_h is in w(i,j,k) and a_i is in w(h,k,j).

3. Core of the method
If there exists only a partial match, then $M = \phi$.
This situation also occurs if some objects are slightly out of place. Since we are interested in partial matches, we introduce the quantity q and modify the iteration formula as follows: $p^{t+1}(i,j)$ = 1 iff p =1 and there exist a subset S of [1,m] with q elements such that $\forall$ s in S, $\exists$ k in [1,n] such that (i,j) $\in$ (k,s).

q is a measure of the way scene and model agree. Setting q to m means that we know that there is a perfect match "a priori." We will denote the resulting set at the tth iteration as M_q^t . The stopping criterion is simply $M_q^{t+1}=M_q^t$. The *main* result is that this process converges in a finite number of iterations. For more detailed analysis, see [10,11].
Some examples of successful application are shown in the section describing results.

4. DESCRIPTION OF THE KERNEL METHOD

One of the main problems with the method described above is that the number of labels in the model has to be small for the method to be efficient. That leads us to selecting a few labels with no really valid criterion. Let us see how we can improve the method if we know for sure that some pairs (i,j) are in the set M .

Let B be a subset of A with q elements, B = { b_i | $1 \leq i \leq q$ } and $B \subseteq A$.
Let T be a subset of L with q elements, T = { t | $1 \leq i \leq q$ } and $T \subseteq L$.
such that all pairs [$(b_i,t_i),(b_j,t_j)$] are pairwise compatible.
Obviously, all these couples are in M_q. Let us call the set of all such couples K_q. Now, a sufficient condition for any pair (a_i,λ_j) to be in M_q is simply that
either $(a_i,\lambda_j) = (b_k,t_k)$ for some k in [1..q]
or $\forall$ k in [1..q] , $(a_i,\lambda_j) \in (b_k,t_k)$

Let us denote this newly obtained set by N_q
$N_q = \{\ (a_i,\lambda_j)\ |\ (a_i,\lambda_j)$ in Kq or $\forall$ k in [1..q], $(a_i,\lambda_j) \in (b_k,t_k)\ \}$

We can see that $M \subseteq N_q \subseteq M_q$, therefore, the new set is better that the previous one.

Finding the Kernel

One of the problems encountered by the relaxation method is that, having to reduce the number of labels, we had to choose some with no valid criterion. Now, to find a valid kernel, we consider the full model and choose a small number of objects in the scene, that is reverse the role of scene and model. The difference is that all labels look alike, but we can choose objects that have distinctive attributes, i.e. they are long, isolated, and correspond to strong edges. The procedure is then:

1. Choose N objects from the scene.

2. Match them with the model using the relaxation method with $q < N$.

Find q matched pairs that verify pairwise compatibility. These q pairs are the kernel.

A flow chart of the procedure is shown in fig. 3.
It is worth noting that finding the kernel as described here is equivalent to finding a complete subgraph of a certain size in a graph, and that this problem is NP-complete. However, the complexity of the problem does not interfere with the process since we control the input size.

Discussion

There are some very interesting properties associated with this method:

1. We no longer need to limit the size of the model.

2. It is a one pass method giving a fast yes/no answer for each object in the scene.

3. The map can be replaced by another scene, such as a different view of the same scene.

4. Since the method to find the kernel is fast, we can "forget" that we know the relative orientation of scene and model and derive it or refine it.

5. RESULTS

These methods have been applied to 2 scenes representing part of the Fort Belvoir Military Reservation in Virginia. The original pictures have been provided by the Defense Mapping Agency and the full resolution images are 2048 * 2048. Figure 4a shows the first view, taken in August, at full resolution. Figure 4b shows the second view, taken in November, at full resolution. Figure 5 is the part of the map corresponding to the previous images. As we can see, the original images are very detailed, and in order to segment them, we proceed hierarchically: To find the most prominent features such as large roads and rivers, we use lower resolution images, as shown on figures 6a and 6b that have a resolution of 256 * 256. Now, as explained in the introduction, we extract the edges, thin them and link them to obtain the linear features shown on figures 7a and 7b. As we can see, most small details have vanished. Since we are interested in roads and rivers, we extract the apars (antiparallel lines) with a maximum width of 8 pixels and filter out the very small ones . The resulting scenes are shown in figures 8a and 8b.

5.1 Relaxation Method

To illustrate the relaxation method, we manually generate from the map a model of the main highway, as shown in figure 9, and match it against the scene of figure 8a. The result is shown in figure 10 and is the desired result.

Once the prominent features are identified in the low resolution image, we can compute better estimates of the scale and orientation and concentrate on the details of the full resolution image, as shown in fig. 11: For each object in the map, such as buildings, we can define a small window in which this object should be, if effectively present. This is illustrated in fig. 12:
fig. 12 (a) shows the segments inside the window where 2 buildings should be,
fig. 12 (b) is the hand made model of these buildings, derived from the map. Their shape is voluntarily inaccurate to show that the method is not sensitive to such small variations.
fig. 12 (c) exhibits all the segments from fig. 12 (a) that are matched with the model of fig. 12 (b).

5.2. Kernel Method

We will use the images at low resolution and apars as primitives. To illustrate the efficiency of the kernel method, we first provide the model from the map as shown in figure 13. This is a full model and contains 35 labels. We use a kernel of 4 elements to match it with the first scene (fig. 8 (a)). The resulting set of matches is shown in figure 14. The processing time was 8.5 seconds, not counting time to compute the kernel. To compare its performance with the relaxation method, we matched the same scene with the full model and a value $q = 9$. The result, shown in figure 15, took 750 seconds, and contains more errors.

We also generated a kernel from the first scene to match the first view with the second view, considered as the model. The second view is a rather complex scene because the original image is very textured, and contains many objects; furthermore, some long segments are broken into small pieces. However, the method was successful, as shown in figure 16.

6. CONCLUSION

This paper demonstrates how a small quantity of "a priori" knowledge can transform a hard problem into a simple one. The "expensive" processing, namely relaxation, is used to find a good match on a small subset of a scene, thus allowing the decision for the other elements of the scene to be simple and fast. This method can be generalized to work on all elements of an image that can be modeled in terms of

vectors in a 2-d space; however, applying it to all the edges of the full resolution image (2048 * 2048) does not appear to be a natural way to proceed, therefore we are currently investigating the existence and representation of intermediate primitive features with a higher semantic meaning than segments:
After the original match at low resolution, we encode the map in terms of these new primitives and match this model with the scene, and only then do we match each primitives with corresponding segments.

7. REFERENCES

1. Azriel Rosenfeld and Avinash C. Kak, Digital Picture Processing, Academic Press, New York, 1976.

2. B. Widrow, "The 'Rubber-mask' technique," Pattern Recognit. 5, 1973, pp 175-211.

3. M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," IEEE Trans. Comput. C-22, 1973, pp 67-92.

4. K. E. Price and R. Reddy, "Matching segments of images," IEEE Trans. Pattern Anal. Mach. Intell., vol PAMI-1, Jan. 1979, pp 110-116.

5. O. D. Faugeras and K. E. Price, "Semantic Description of aerial images using stochastic labeling," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-3, Nov. 1981, pp 633-642.

6. S. A. Dudani, A. L. Luk, J. P. Stafsudd, C. S. Clark, B. L. Bullock, "Model based scene matching," Hughes Research Report 509, Malibu, 1977.

7. C. S. Clark, A. L. Luk, and C. A. McNary, "Feature-based scene Analysis and model matching," Proceedings of NATO Advanced Study Institute on Pattern Recognition and Signal Processing, Paris, Ju e 1978.

8. C. S. Clark, W. O. Eckhardt, C. A. McNary, R. Nevatia, K. E. Olin, and E. M. VanOrden, "High accuracy model matching for scenes containing man-made structures," Proc. of Symposium on Digital Processing of Aerial Images, SPIE, vol. 186, 1979, pp 54-62.

9. R. Nevatia and K. R. Babu, "Linear feature extraction and description," Computer Graphics and Image Processing, vol. 13, 1980, pp 257-269.

10. G. G. Medioni, "Matching of a map with an aerial image," USC-IPI Report 1050, Sept. 1981, pp 12-37.

11. R. M. Haralick and L. G. Shapiro, "The consistent labeling: Part 1," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-1, Apr. 1979, pp 173-184.

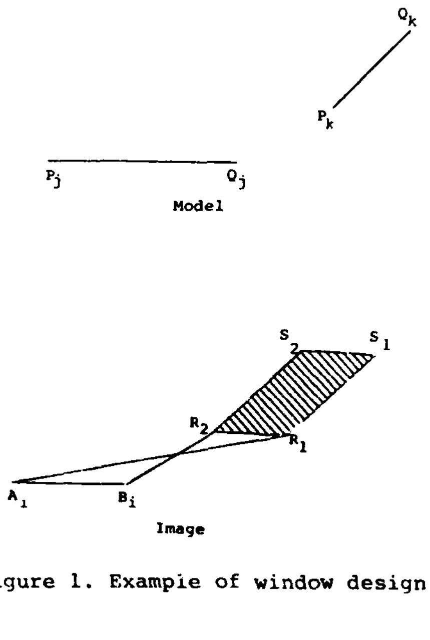


Figure 1. Example of window design.

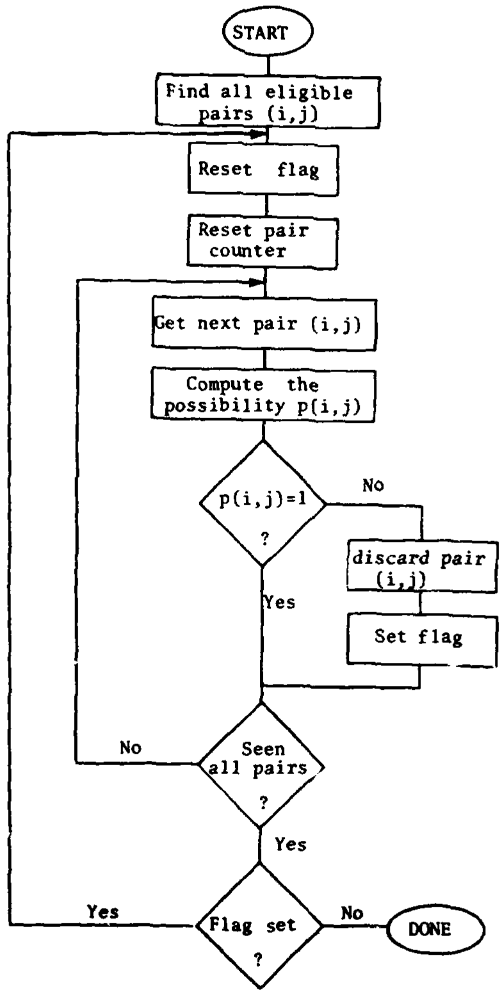


Figure 2. Flow chart of the relaxation method.

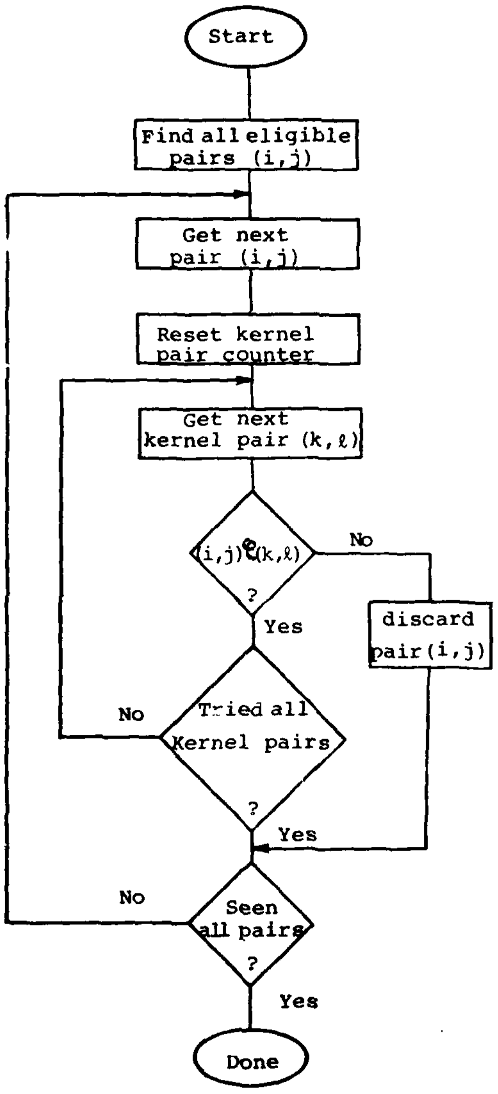
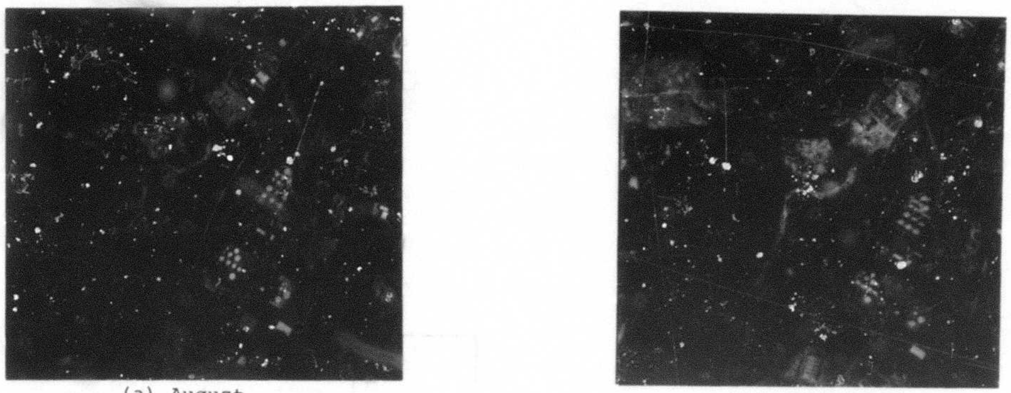


Figure 3. Flow chart of the kernel method.


(a) August

(b) November

Figure 4. Full resolution view (2048 x 2048)

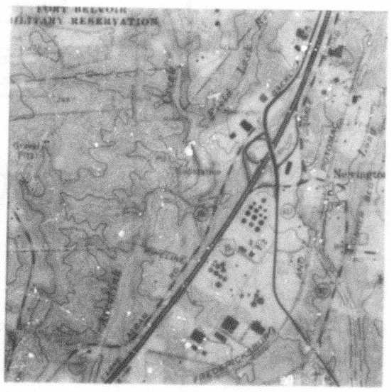
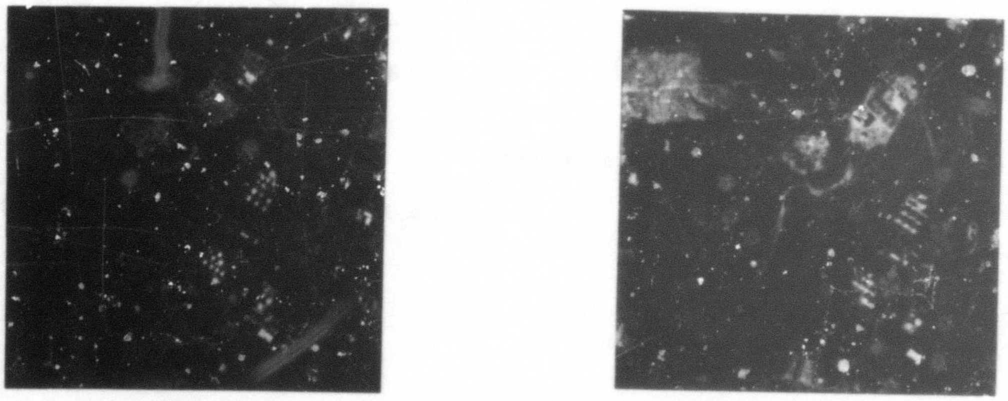
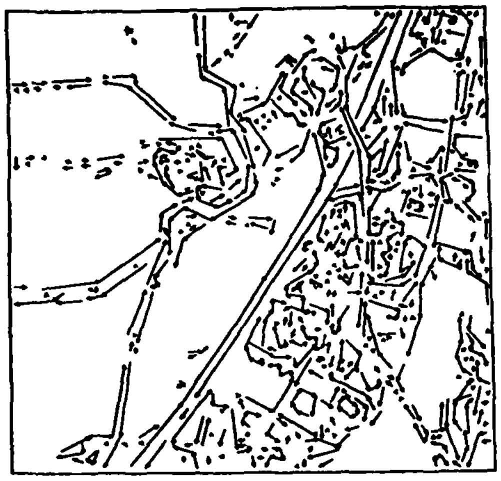


Figure 5. Map of the area


(a)August

(b) November

Figure 6. Low resolution view (256 x 256)


(a) First view

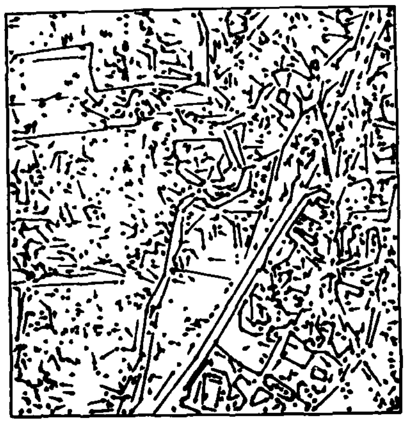
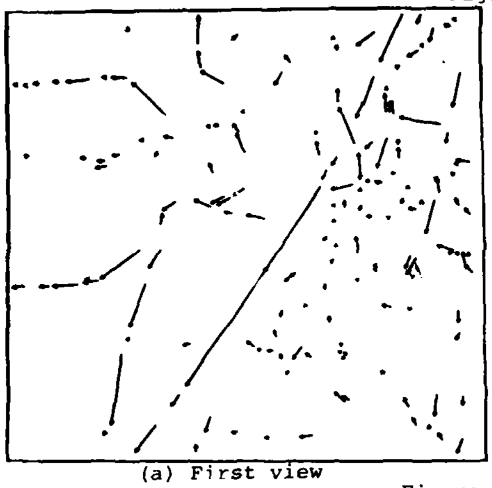

(b) Second view

Figure 7. Segments


(a) First view

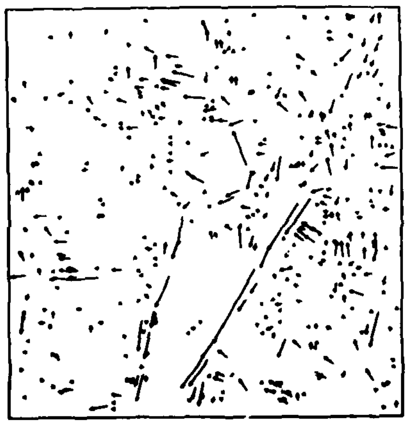

(b) Second view

Figure 8. Antiparallel lines

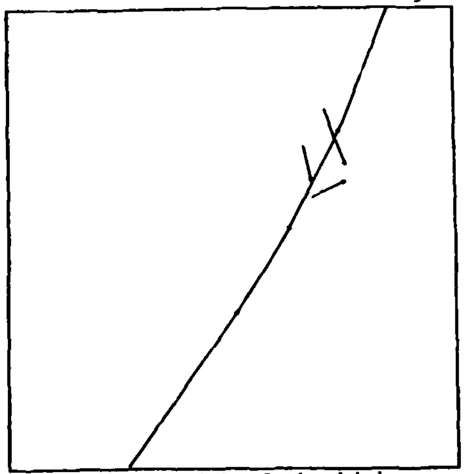


Figure 9. Model of the highway.

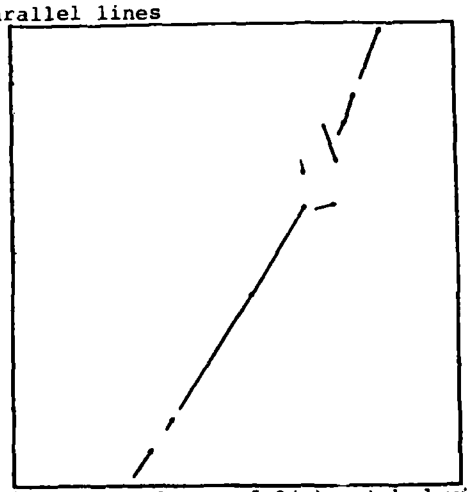
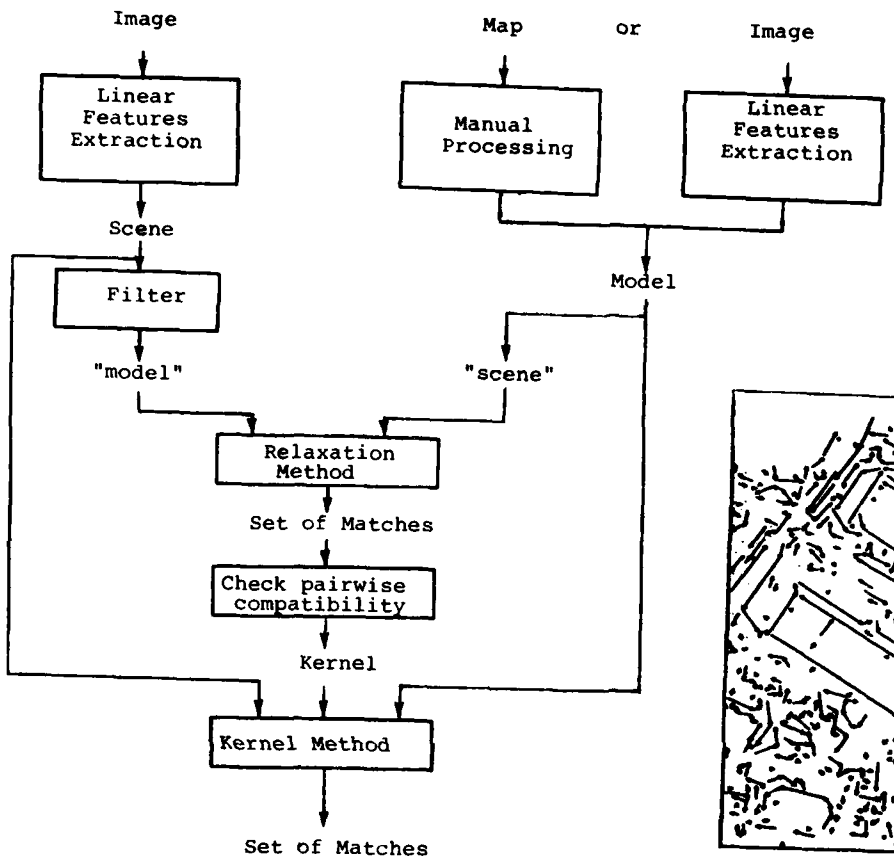
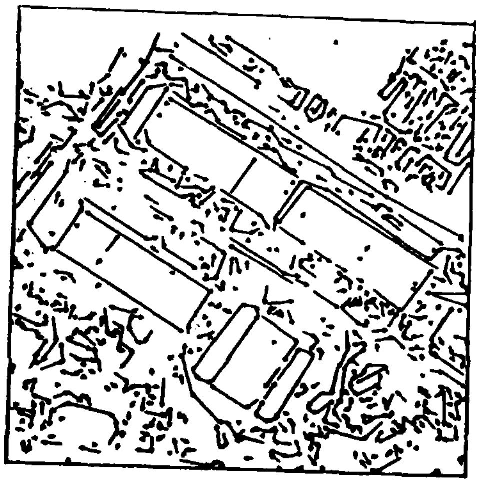


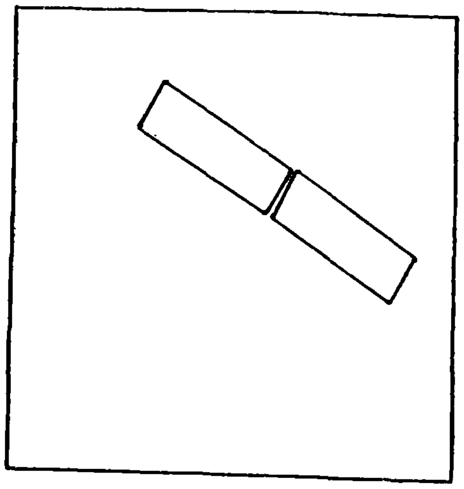
Figure 10. Apars of 8(a) matched with the model of Fig. 9.



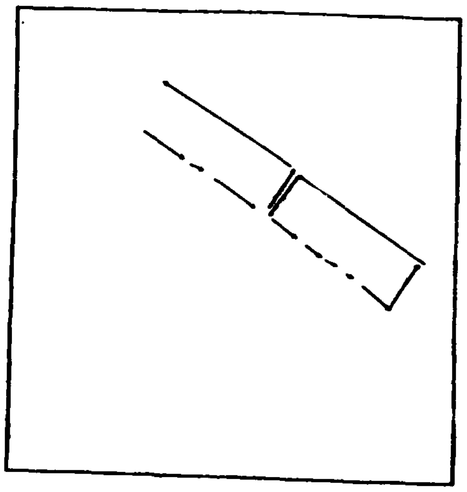


(a) segments

Figure 11. Overall Processing



(b) model



(c) result

Figure 12. Matching buildings

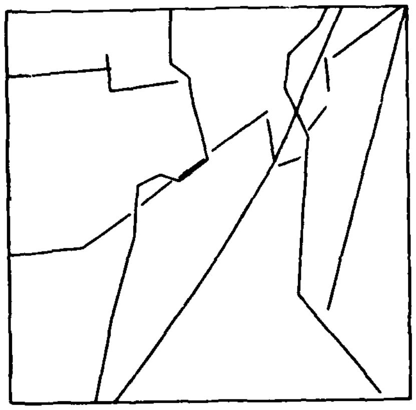


Figure 13. Full model from the map

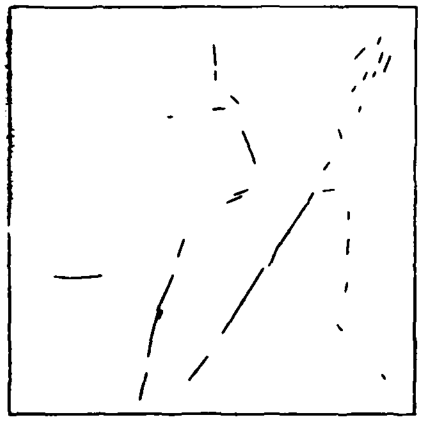


Figure 14. Apars of Fig. 8(a) matched with the model of Fig. 13 using kernel method.

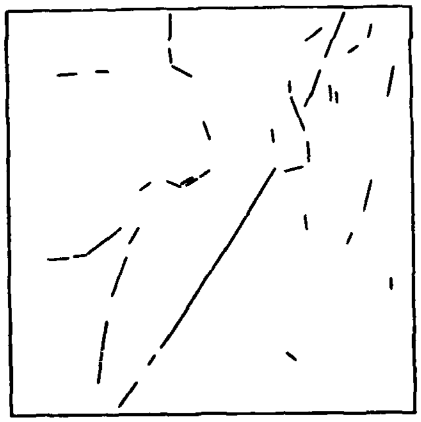


Figure 15. Apars of Fig. 8(a) matched with the model of Fig. 13 using relaxation method.

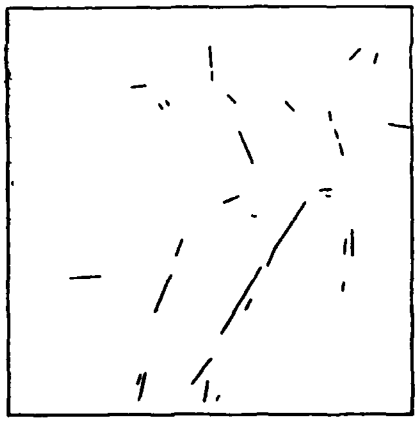


Figure 16. Matching 8(b) with 8(a) as the model.

AD-P000 117

REPRESENTATION OF TIME AND SEQUENCES OF EVENTS

Jitendra Malik and Thomas O. Binford

Computer Science Department
Stanford University, Stanford, California 94305

Abstract

This paper describes a *new approach to representing and reasoning with temporal information. A wide variety of temporal specifications can be converted into linear inequalities relating the endpoints of the events. Linear programming is then used to represent these constraints and perform deductions. The information is modularized into semantically related clusters of events each with its own tableau and related to each other by a reference frame transformation. This provides a uniform formally adequate representation which is complete and also computationally efficient.*

Introduction

Most work in artificial intelligence which deals with *real world* problems would require some reasoning with time. The importance of such a temporal understanding in the areas of problem solving and natural language understanding has been recognized earlier[1, 5]. In image understanding applications too, it is necessary to know about sequences of events in order to identify a dynamically changing scene.

Most problem solving systems have modelled time using a state-space approach. In this approach the world is described as a sequence of snapshots each with a set of facts holding at the time instant. Because of the inadequacy of this approach, attempts have been made to incorporate time explicitly in planning [3,6,5]. In particular Vere's DEVISER [5] is a general purpose planner which generates parallel plans to achieve goals with imposed time constraints. Both durations and start time windows for sets of goal conditions may be specified. The parallel plans consist of not just actions but also of events(triggered by circumstances), inferences, and scheduled external events (completely beyond the actor's control).

However Vere's system is not a general purpose temporal reasoning system. Time is incorporated into the planner with the notion of a window -which is typically an upper and lower bound on the time when an activity may occur. Windows are specified explicitly for goals and are computed dynamically during plan generation by considerations of durations of intervening activities and the times of occurence of scheduled external events. This is done by window revision algorithms which push activities forward or backward on the time-line when they get ordered or when the durations become known. The temporal representation and reasoning is ad hoc and tied to the needs of the planner.

A more general purpose approach is that taken in the systems [1,2,4] that build time specialists. Such a subsystem maintains temporal relations and provides the rest of the system with tools to store, retrieve, delete and reason with the temporal information. There are two major requirements for a time specialist: First, it must be *formally adequate*, and second it must be *computationally effective.* The first condition is met if the formal system is coherent and consistent, and contains sufficient mechanisms to be able to represent all temporal specifications and perform all the deductions we want. The second requirement is essential in order to have the program produce answers with a reasonable amount of effort. This paper describes another attempt in this direction.

In the next section we develop the notion of temporal specification. The following section describes the representation scheme. Lastly we describe the deductions that can be performed by the reasoning module.

Temporal specifications

A temporal specification[4]is a statement that partially specifies in some manner, the time of one or more events. Examples are:

(1) The gas leak started immediately after takeoff.
(2) John saw Mary a while ago.
(3) My fever lasted 3 days.
(4) A few days back, I was in Las Vegas.
(5) Hiroshima was bombed on August 6, 1945.
(6) I will finish my PhD in two or three years.
(7) Jack had an accident a month after getting to Boston.

The next three subsections attempt to clarify different aspects of this. However, it may be observed that except for (2) and possibly (4) all the specifications are linear relations between the endpoints of events.

1. Modelling uncertainty.

As the examples indicate temporal information is often incomplete and imprecise. However one can draw a distinction between two kinds of incompleteness.

(a)Underconstrained in the mathematical sense. When we say that event B occurs after event A, the associated information is expressed in the inequality $start_B \geq end_A$. There are an infinite number of pairs of values for $start_B$ and end_A that satisfy this inequality. However the mathematical meaning of this statement is absolutely precise.

(b)Linguistic fuzziness. When we say that "Jane saw the doctor a few days ago " the problem is that of translating from the natural language phrase to an assertion in the formal representation used. Several approaches [7] can be taken:

1. **Uncertainty intervals**— Some cases of vagueness can be thought of as a value at an indeterminate point in a well-defined interval.The phrase **Last year** means a point somewhere in the interval **12:01 am January 1, 1981 to 12:00 midnight, December 31, 1981**. The phrase **a few days ago** can be defined (arbitrarily) as between 2 and 10 days.

2. **Plausibility distributions**— It may be better to think of an imprecise term as being more or less applicable over a range of values. The phrase **a few days ago** would be very applicable exactly 3 days ago, moderately applicable 7 days ago, and highly inapplicable 1 year ago. The degree of applicability can be quantified by using probabilities and manipulations done using fuzzy theory.

3. **Linguistic variables**-- Instead of converting imprecise phrases to numerical measures, one leaves them in a symbolic form and use *ad hoc* rules to manipulate them. This approach is used in Kahn and Gorry[4] who would translate a sentence **John will finish his thesis in a few months** into a time-expression of the form **(AFTER(ALL-OF TODAY (FUZZY-AMOUNT (NIL A-FEW MONTHS)**. The structure for a **FUZZY-AMOUNT** includes a quantifier (*eg* **ABOUT, NEARLY)** and a fuzzy-number (*eg* **A-FEW,SEVERAL**).

In our system, the uncertainty interval approach is used. No attempt is made to translate phrases of the type **a few days ago** as it is believed to be primarily a problem of natural language understanding.

2. Time Points vs Time Intervals.

Temporal information can be given both as relations between endpoints and between intervals. A fact like **The bank opens at 9:00 am** is a statement about the startpoint of the event **bank-is-open**. A fact like **The Cuban missile crisis took place during Kennedy's term** is a statement about intervals. The two approaches are equivalent in terms of representational power. Any statement about intervals can be converted to an equivalent statement about the endpoints of the associated events eg

$$A \ \ before \ \ B \leftrightarrow end_A < start_B$$

If one is using an interval-oriented approach, endpoint information can be represented by associating with each event two instantaneous events—one corresponding to the start of the parent event and the other to the end.

For the convenience of the user, both forms of specification are allowed. The internal representation is in terms of endpoints as that is more natural for our representational scheme.

3. Absolute vs Relative: Changing Reference frames.

The everyday notion of time is relative rather than absolute. We remember events in semantically related clusters revolving around a key event. Even time as measured by the Gregorian calender is relative rather than absolute. This suggests an organisation for the temporal information—semantically related clusters each of which represents information about the events in its own reference frame. As in the case of spatial information, it then becomes necessary to provide for transformations to relate the time coordinates of events in different reference frames. The transformation is linear.

The representation

As may have been observed, all the temporal specifications are equivalent to linear relations between the endpoints of the events. This means that we can use linear programming to represent and reason with temporal information. A time specialist based on linear programming is guaranteed to be formally adequate --unlike the *ad hoc* methods. It provides a uniform representation for storing the wide variety of temporal information. This is a major change from the philosophy of earlier systems. Kahn and Gorry[4] use several different ways of organizing the events—with a date-line, using before/after chains, and using special reference events each with a separate procedure for making deductions. Allen[1] uses a network of constraints to maintain all possible relationships about how the intervals in it are related. However, in his system no metric information is represented and thus fails to be formally adequate by our criteria.

To people conditioned to react with horror to uniform formally adequate schemes, our representation would immediately raise the specter of inefficiency. Indeed, this would be so if all the temporal facts about the domain were to be represented in the same linear programming tableau. Recall, however that we can organize the information in semantically related clusters—each with only a small number of constraints. The system still remains complete because we can do a reference frame transformation to relate events in different clusters. This idea buys us the same advantage as the reference interval concept[1,4] in a more systematic way. The analogy with the way we organize and reason with spatial information suggests the naturalness of this approach.

Now for the details—the linear programming is done using the simplex algorithm in the version formulated by Tucker. In this approach the rows of the tableau have a direct physical meaning—they correspond to the endpoints of the events. The system was written in MACLISP in the ACRONYM environment so that it could be used easily as a module for future image understanding work.

Temporal Reasoning

As the tableau represents all the information in the temporal specifications, the system is complete —all deductions that can be made from the constraints can be made from the tableau. The general approach is to formulate an expression which is maximized or minimized, while still satisfying the constraints in the tableau. The implemented features include

1. **Satisfiability**—As the linear constraints associated with each temporal specification are entered into the tableau, the existence of a feasible solution is checked. The system refuses to accept a constraint that is inconsistent with the previous set.

2. **Bounds**— One can determine the upper and lower bounds for any variable, which corresponds to an endpoint of an event, or a linear expression in these variables. For example, this permits us to find upper and lower bounds on the duration of an event.

3. **Possibility and Necessity**—. If a predicate's being true would not be inconsistent with the constraints in the tableau, the predicate is said to be *possible*. If a predicate's being false would be inconsistent with the constraints of the tableau, the predicate is said to be *necessary*. These deductions would be useful to a planner using this time specialist. If event A is necessarily after B, then that ordering can be done right away. Possibility considerations can help prevent unnecessary backtracking.

Acknowledgements

This work was supported under ARPA contract N00039-82-C-0250.

References

[1] Allen,James F.,"An Interval-Based Representation of Temporal Knowledge," *Proceedings IJCAI-7*, (1981), 221-226.

[2] Bruce,B.,"A Model for Temporal References and its Application in a Question Answering Program," *Artificial Intelligence*, 3 (1972), 1-25.

[3] Hendrix,G.G.,"Modeling Simultaneous Actions and Continuous Processes," *Artificial Intelligence*, 4 (1973), 145-180.

[4] Kahn,Kenneth and G. Anthony Gorry,"Mechanizing Temporal Knowledge," *Artificial Intelligence*, 9 (1977), 87-108.

[5] Vere,Steven A.,"Planning in Time: Windows and Durations for Activities and Goals," Technical Report, Jet Propulsion Laboratory.

[6] Wesson,R.B.,"Planning in the World of the Air Traffic Controller," *Proceedings IJCAI-5*, (1977), 473-479.

[7] Winograd,Terry,"Language as a cognitive process,"Readin Addison Wesley.In preparation.

AD-P000 118

Modeling The Sequential Behavior of Hough Transform Schemes

Christopher M. Brown
David B. Sher
Computer Science Department
University of Rochester
Rochester, NY 14627

Abstract

The Hough transform provides a paradigm for multi-dimensional pattern detection and more general parameter extraction. It has good properties in noise, but as usually implemented requires an accumulator array whose size is exponential in the number of parameters extracted. The idea of accumulating votes in a small content-addressable store raises many technical issues, some of which are outlined here. Simulation results illustrate some of the points.

The preparation of this paper was supported in part by the Office of Naval Research under Grants N00014-80-C-0197 and N00014-82-K-0193, and in part by the Defense Advanced Research Projects Agency under Grant N00014-78-C-0164.

1. Introduction

The Hough Transform (HT) maps a point of one parameter space A (often a local feature space or generalized image) into a set of compatible points of another space B (often a space of "higher-level," more global parameters) [Hough 1962; Duda and Hart 1972; Ballard 1981]. This process is sometimes called "voting" (the feature points vote for possible higher-level constructs of which they could be a part).

HT is traditionally implemented by accumulating votes in a quantized version of the parameter space B, called an accumulator array. A naive implementation of the accumulator array uses space exponential in the number of dimensions (parameters)--this makes multi-parameter HT impractical. One approach to this problem is to quantize the parameter space dynamically [Sloan 1981; O'Rourke 1981]. Another idea, the subject of this report, is the cache-based HT, which uses a fixed (small) content addressable store (in software, a hash table; in hardware, a cache) to accumulate votes.

To study the properties of the cache-based HT, it is helpful to have an abstract, domain-free characterization of the HT vote-generation process. Existing applications (e.g., [Brown and Curtiss 1982]) and theory (e.g., [Shapiro 1975, 1978a, 1978b; Brown 1982]) suggest considerations that should be taken into account by an abstraction. The abstraction may be analyzed or simulated with various cacheing schemes to derive a model of cache-based HT performance. Then, existing applications (such as our shape, motion, and illuminant-direction computations) may be used to test model predictions.

Some considerations for a general model of the cache-based HT process appear in Sections 2 and 3, and Section 4 gives an abstraction of the voting process. Section 5 outlines some possible analytical approaches. Section 6 presents results of pilot studies on cache behavior. Figure 1 is a diagram of the system for analysis of sequential behavior of HT schemes.

<<Figure 1>>

2. Informal Abstract Hough Transform

This section lists considerations that govern the behavior of any HT-like process emitting votes through time into an accumulator. Such an abstract process is called a voting machine in Section 4.

In a noiseless HT, one feature point in A produces a set H of votes that generally are weighted points in the parameter space B (so a vote is a vector (w, b), with w a weight and b in B. (We assume here that the HT is "tuned" to one parameter transformation, to detect instances of one shape, etc.) In one limiting case, the set H is unbounded (as when a point on a line votes for an infinite line of points in (slope,intercept) space). In another limiting case, H is a singleton (as in the work of Shapiro [1975, 1978a, 1978b]). In either case, for noiseless HT we assume that H from a feature point in A contains exactly one vote for the "true" parameters in B that characterize the instance in A. Let us call the true parameter value the peak or mode for the instance, since that is where the plurality (mode) of votes for it should be. H is the feature point spread function (fpsf) of [Brown 1982]. In all that follows, we assume accumulation into a quantized version of B, so H = fpsf is a discrete set. Of course H may be a complicated function of the parameter values and locations in the originating parameter space A: H = fpsf = fpsf(x, A(x)). As A is scanned, individual feature points cause votes to be emitted. The summation of all H from an instance into the accumulator array is the parameter point spread function (ppsf).

(Consideration 2.1) Instances in an image consist of several feature points, extend across several scan lines, and are spatially coherent. There may be more than one feature on a scan line. Instances may overlap.

(Consideration 2.2) The image can be scanned in any of several ways: e.g., left-right, top-bottom; pixels at *random (with and without replacement)*; left-right across scan lines in random order; left-right across interlaced scan lines.

(Consideration 2.3) Votes come in bursts of length $||fpsf(x, A(x))|| = ||H||$, each generated by a single feature vector in A.

(Fact) With one instance and no noise, successive bursts all come from the single instance. With multiple instances and no noise, successive bursts may or may not come from the same instance. The likelihood that they do depends on assumptions about the distribution of instances in the image (do they overlap?) about the distribution of feature points in an instance (is there more than one per scan line?), and about the scanning pattern. (The basic question is whether two features of the same instance are encountered before another instance.)

(Consideration 2.4) Only one (weighted) vote in H is a vote for the true instance parameters (for the peak or mode).

(Consideration 2.5) A basic question in HT is its performance in noise. H's from noise points are interleaved with H's from instances. There are several noise phenomena with differing characteristics and implications. First, there is statistical noise. One form is signal-independent image degradation. This process establishes a kind of baseline, since its random nature makes it one of the least pernicious forms of noise in the cache-based HT. One component of degradation, perhaps worth singling out, is that component consisting of features not associated with any instance in A. This component adds extraneous H's, but does not affect H's from true instances.

Another form of noise is errors in computing "image features," i.e., the feature vectors in A. These errors, as well as location and quantization noise in A, may be modeled as certain random processes.

The most pernicious form of noise, and one not treated here, is systematically misleading instance information. For instance, extreme and systematic feature dropout can arise from occlusion. Especially troublesome are near-miss instances (many features in common with instances of interest). The problem is that in the cache-based HT, it is important to create and retain peaks for the instances. These peaks are threatened by competition from other growing peaks, such as arise in near-miss instances.

3. Informal Abstract Cache

A hash table or cache accepts one element at a time, is content-addressable, and garbage collection, or flushing, happens when its finite length is exhausted. Unlike a cache for memory management, the HT cache is not meant to track a dynamic situation, but to function as a "smart accumulator."

Our cache is a data structure and associated operations for management of a set S from the space of weighted vectors from B. Thus elt = (w, b) for some $w \in$ Reals (or $w \in$ Integers) and $b \in B$. A cache has a parameter called its length--the number of elements it can hold. As do most abstract data types dealing with sets, a cache has basic initialization and housekeeping operations, as well as operations of the following flavor (these are for exposition, not practicality).

Full(cache): TRUE if cache full, else FALSE.
FindVector(searchelt,cache, @foundelt): sets foundelt to unique element whose vector agrees with that of searchelt and returns TRUE, or returns FALSE if no such element is in cache.
FindVote(weight, cache, @foundeltSet): sets foundeltSet to set of elements whose weight agrees with (or is less than or equal to) weight and returns TRUE, or returns FALSE if no such element is in cache.
FindMax(cache, @foundelt): sets foundelt to the element (or one of the elements) with maximum weight.
Remove(elt, cache): removes elt from cache.
Insert(elt, cache): adds elt (not in cache) to non-Full cache (see Replace).
AddWeight(elt, cache): does nothing but return FALSE if no element matching elt's vector is in cache. Otherwise it increments the matching vector's weight by the weight of elt and returns TRUE.

There are three high-level cache-management operations: Vote, Flush, and Interpret.

Vote(elt, cache): Vote uses AddWeight to increment the weight (vote count) of an existing element, or else Inserts element elt into the cache if there is room, or as a last resort invokes a more or less complicated method of dealing with a full cache, usually involving Flush and possibly re-trying the Insert.
Flush(cache): Flush creates room for more elements. Its design is a focus of current research (Section 6). Algorithms with hardware implementation are especially of interest.
Interpret(cache): Interpret is to identify "peaks" in accumulator space B, which it is hoped correspond to (are the parameters of) instances in the input space A. If the input contains only a single instance, the natural Interpret returns the element of maximum weight (most votes). For multiple instances, especially an unknown number of them, Interpret presents interesting problems, and is a focus of current research. As with Flush, algorithms with a natural hardware implementation are especially of interest.

(Consideration 3.1) The Cache has a fixed length. If that is infinite, the cache acts like the accumulator array.

(Consideration 3.2) The Cache is often most easily addressed as a linear data structure, with vote vectors transformed to offset integers. Addressing modes may vary in hardware implementations.

(Consideration 3.3) There must be a cache-flushing, garbage-collection strategy for dealing with overflow. The obvious basic strategy is to flush entries with fewest votes. There are many variations, including using recency information (traditional for LRU caches) and geometric locality information (implemented through a hierarchical cache arrangement).

(Remark) For applications, flushing strategy may reflect considerations of hardware implementation.

(Remark) The interaction of CHough [Brown 1982, Brown and Curtiss 1982] with cache-based HT raises interesting questions, since it involves negative votes.

4. A General Voting Machine

A geometric imaging-like situation is an easy way to describe the voting machine, and in fact the most straightforward implementation probably is as shape detection applied to an image-like array. A description as some form of stochastic FSM would work equally well, but isn't as visualizable.

The abstract image is a two-dimensional array A(x) of vector entries corresponding to instances and to noise. The voting machine scans this array, and for each feature vector it encounters it emits an H = fpsf(x,A(x)). Houghing for a global parameter (like sun angle) is modeled by an A(x) with only one instance. Houghing for multiple instances is possible. Noise (Consideration 2.5) may be modeled.

The following technical decisions arose from geometric encodings of the considerations in Section 2 and from [Brown 1982].

1) Let an instance be one or more vertical lines (called parts) in a horizontal row, separated by empty columns. This captures all facets of Consideration 2.1.

2) Various forms of scanning (Consideration 2.2) are implemented by accessing A in different orders.

3) The fpsf's for an instance should look like diameters of a circle (Figure 2). The HT of an instance (the contents of the accumulator array) will then be a radial rosette of lines all passing through a central point, the peak. This decision is based on Considerations 2.3 and 2.4, and the fact that this choice reproduces the ubiquitous near-peak 1/r falloff of parameter psfs in all dimensions [Brown 1982].

<<Figure 2>>

4) Noise can take various forms (Consideration 2.5). Missing instance points can, if systematic, lead to non radially symmetric ppsfs. Noise feature vectors can be added into the A(x), and of course instance feature vectors can be perturbed by noise processes. In the experiments reported here, we had no massive dropout (modeling occlusion), getting a similar effect by varying the number of parts per instance. We did not attempt to address questions of near-miss instances at this stage. For this report we used the baseline case of signal-independent image degradation.

The variables that characterize the abstract voting machine are:

V1) ||fpsf||
V2) number of instances and their location
V3) number of parts/instance
V4) noise parameters of three types
V5) scanning method

Figure 3 shows an A(x) with two 2-part instances with and without noise. Each part is 9 features long; there are eighteen different features.

<<Figure 3>>

5. Analytic Approaches

1) Hypothesis Testing: After the HT, the accumulator holds a distribution of votes. After cacheing the same data, the cache is meant to mirror certain characteristics of the accumulator's distribution. In particular, it is meant to have the same modes (local maxima). One could proceed with something like a statistical hypothesis testing approach, with the null hypothesis being that modes in the cache correspond to modes in the accumulator. The idea of course is to measure the probabality of Type I and Type II errors (falsely rejecting and falsely accepting the null hypothesis), given some sample statistic--in our case, the cache contents. Usually, we choose probabilities α and β for these errors that are small enough for our purposes. When things go well (i.e., in textbooks) this approach yields a critical (rejection) region of values of the test statistic for which we should reject the null hypothesis if we are to maintain our standards for error probabilities. There are non-parametric tests for things like the sameness of median between two distributions.

What goes wrong immediately here is that the sampling performed by the cache is extremely structured. Even if we were to attempt to compute with the known relevant distributions [Brown 1982], the intricacies of this sampling seem very resistant to analysis.

A generalization of the hypothesis testing approach is a decision theory approach, that allows a more flexible cost function to be associated with decisions. The appealing thing there is perhaps that it can more respectably be based on empirical data. Sequential decision theory seems possibly more relevant, though less well understood.

2) Page Replacement, etc.: There is a long tradition of work (e.g., the analysis of page replacement) aimed at characterizing cache behavior. Almost certainly none of that work is directly relevant, but it may provide useful analytical tools and abstractions.

3) Geometry and Probability: Last (this seems most promising), the particular abstraction chosen to model cache-based HT might be analyzed with very straightforward probability and statistical tools. The model proposed in Section 4 is very geometrical in flavor, corresponding to throwing line segments sequentially down on the plane and analyzing their accumulating statistics of overlap. It may well be that basic, geometrically based probability theory is directly relevant.

6. Results

6.1 Overview

We implemented the abstract voting machine of Section 4 and a software simulation of a cache that is parameterizable in several ways, including length and flushing strategy. The cache is instrumented for various forms of statistical analysis. The results are presented in tables and figures that follow this section. They are preceded first by a high level overview of the experiments, then a more detailed explanation of the parameters involved.

We ran a set of experiments to explore several parameters (both of the cache and of the abstract image) governing sequential HT performance.

Fixed Parameters:	Flush Algorithm (Threshold)
	Number of Instances (1)
	Part Length (11)
Varying Parameters:	Cache length (32, 64, 128, ∞)
	Number of Parts (1, 2, 3)
	Noise (0, 15%, 30%)
	Fpsf Length (3, 9, 15 cells)
Scanning method	(five methods)

The results of these experiments are summarized in Tables 1-4, which show a "peak/background ratio" measure and its variation for each case. (Tables 1, 3, and 4 appear in [Brown and Sher, 1982].) The ratio is computed as follows.

$$\text{Ratio} = \frac{\text{weight of vector describing instance}}{\text{maximum weight of all other vectors}} \qquad (1)$$

<<Table 2>>

Thus a ratio of 3.00 means that the single parameter vector correctly describing the instance is three times as high as its nearest competitor. A ratio of unity means there is a tie. Any ratio greater than unity means the correct peak would be found by simply choosing the highest peak. A ratio less than unity means such simple thresholding finds a vector giving an incorrect instance description.

We expanded a subset of these results in more detail.

Fixed Parameters:	Flush Algorithm (as above)
	Number of Instances (as above)
	Part Length (as above)
	Number of parts (1)
Varying Parameters:	Cache length (as above)
	Noise (15%, 30%)
	Fpsf Length (as above)
	Scanning method (two methods)

The expanded results are shown in Figure 4, which gives the histograms of the peak/background ratio whose means and standard deviations appear in Tables 1-4. From the histograms it can be seen, for instance, what fraction of the time an instance would not be detected by a simple thresholding version of the Interpret function.

<<Figure 4>>

Finally, Figure 5 shows histograms like those of Figure 4. Here the main focus is on the Flush algorithm as cache length and noise level vary.

<<Figure 5>>

6.2 Details

The choice of which parameters to vary and interesting values for them was determined by a significant amount of prior experiment. The size of the abstract image is irrelevant, being redundant with other parameters such as noise density. In fact the image was 20 x 20, and the single instance was located in its center.

Flush Algorithms: We used Threshold Flush exclusively in the main experiment. It simply raises a threshold for the vote count (weight) of elements until it finds elements whose weight is at threshold, all of which it flushes. For the last experiment, we compared Threshold Flush with Random Below Threshold Flush, which simply finds an element at random whose weight is in the bottom nth percentile of weights and Removes that single element. In this experiment the doomed element was picked at random from the bottom third of weighted elements.

Number of Instances: Only one instance appears in images for these experiments. This is an important case, and fairly terse statistics describe the efficacy of the simple-threshold Interpret algorithm. Multiple instance performance is a matter for future research (also see [Brown and Curtiss 1982]).

Number of Parts: One-, two-, and three-part instances generate (in the noiseless case) that number of feature points per horizontal scanline.

Part Length: Part length 11 means the instance extends across 11 horizontal scan lines. The total "signal strength" for the instances is 11, 22, and 33 votes.

Cache length (32, 64, 128, infinite): The infinite cache is a pure accumulator array. This variable can to some extent be traded off against Fpsf Length. It is worth mentioning that commercially available hardware has cache lengths on the order of thousands of words.

Noise (0%, 15%, 30%): These experiments use a noise model that mimics signal-independent image degradation. A random feature vector overwrites an abstract image point (instance or background) with the indicated probability.

Feature point spread function (fpsf) Length (3, 9, 15 cells): Short fpsfs produce less inherent noise (fewer sidelobes) [Brown 1982]. For example, consider line detection: As better and better edge gradient information is incorporated to reduce the possible range of slope attributed to the line, the fpsf shrinks. Inherent noise is not a problem in single-instance images, but the effects of the statistical Noise are also felt at greater distance with larger fpsfs, and that accounts for the performance degradations noticable here. 9-cell fpsfs were used in Fig. 2b.

Scanning method: We use five methods to scan the abstract image. The first four hit each image point exactly once, the last is not guaranteed to. The first two are deterministic, the last three are random. The first, fourth, and fifth might be easily implemented in hardware.

1) Left to Right within Top to Bottom: as in a non-interlaced TV scan.
2) Top to Bottom within Left to Right: interesting because it causes several fpsfs from the instance to arrive almost successively.
3) Random without replacement: implemented by accessing the 400 elements of the array in permuted order.
4) Left to Right within randomly permuted scanlines.
5) Random with replacement: Access n^2 elements of the nxn image array at random, very likely missing some entirely and consequently hitting others more than once.

N: There are two sources of randomness: One arises from noise, one from the random scanning methods. We ran ten different noisy images into each scanning method. The random methods each produced ten different orders of scan, the scanline methods each produced one, which was repeated ten times. Thus in the finite cache experiments, the N for the two non-random scanning methods (1 and 2) is 10, and for the three random scanning methods (3, 4, 5) is 100. In the infinite cache, the four methods (1,2,3,4) that hit each element once are equivalent, so are collapsed into one case.

6.3 Conclusions and Future Work

How should these results be interpreted? To apply them to any particular HT situation (but so far only to detection of a single instance or derivation of a parameter vector for a single phenomenon, and only with signal-independent noise), analyze it in terms of scanning algorithm, number of feature points, noise level, fpsf length and so forth. Interpolate or extrapolate as necessary. Remember that the important basic process is the succession of votes emerging from the HT (its mix of feature and noise elements). The explicit conversion from any given HT situation to parameters of our model will be addressed in more detail in a future report (or a later edition of this report).

The infinite cache (a pure accumulator array) is not dramatically better than the finite caches under the circumstances of these experiments. Thus finite caches seem to be practical HT accumulators.

The tables and histograms show a graceful and qualitatively predictable degradation of performance as noise and fpsf length increase and cache length decreases.

The random (with replacement) scan was uniformly significantly worse in all cases than other methods. The random (without replacement) does not seem significantly better. In fact, the results are (perhaps surprisingly) insensitive to scanning method, as long as each input space point is hit once.

Even in 30% noise, the Top to Bottom within Left to Right scanning method was not better than the Left to Right within Top to Bottom. This indicates that successive bursts of votes (fpsfs) containing votes for the "true parameter" are not enough to help the peak establish itself. Probably the noise that arrives before the instance and the flush algorithm dominate the peak-establishment process.

Several of the histograms show a significant bimodality with one peak at 0. This indicates that the "true peak" never gets established in a significant number of cases, and that some pre-processing might help dramatically in establishing peaks. One idea is "pair Houghing," in which only vectors that appear in two fpsfs are allowed into the cache. Clearly without such preprocessing the issue is whether peaks are able to survive in the cache, and this depends to a large extent on the Flush algorithm. Instances found early in the scan might have an advantage in some flushing strategies, while other strategies might penalize peaks that have been in the cache longer. We reasoned that the simple Threshold Flush might be systematically flushing nascent peaks before they got established, thus accounting for the significant number of 0 peak/background ratios.

Jerry Feldman suggested the Random Below Threshold Flush to replace the systematic "slaughter of innocents" imposed by Threshold Flushing. The result is a lottery-like system that selects from a range of vote weights, allowing some small peaks to survive. As shown in Figure 5, this method makes for a significant improvement in mean peak/background ratio, and reduces the peak at 0.

Systematic distortion, near-miss instances, and occlusion pose thorny problems for the cache-based HT. We emphasize that many of the conclusions of this report and all the quantitative data are based on a statistical, signal-independent noise model. An important topic for future work is to assess the performance of cache HT schemes in these more stringent noise conditions and to develop means of improving such performance.

In CHough methods [Brown 1982, Brown and Curtiss 1982], the fpsf contains negative as well as positive votes. CHough interacts in an interesting way with flushing strategies. Future work will address these issues.

Hierarchical caches can incorporate a form of "geometric locality" into flushing. Natural flush algorithms for a single-level cache are based strictly on weights. Multiple caches in a resolution pyramid, maintained in parallel, can be used to restrict flushing to coherent localities in the input space A. This idea is another matter for future research.

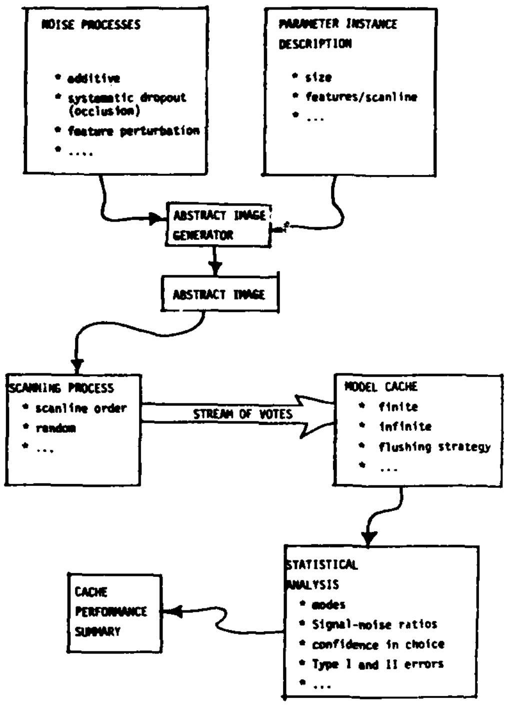
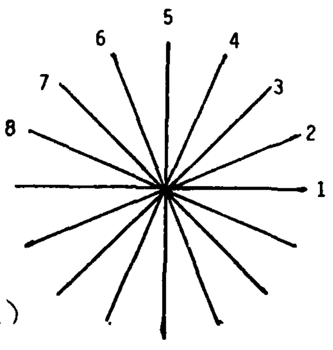


Figure 1: The cache-based Hough transform analysis system.



(a)

(b)

-	-	-	-	-	-	-	-	-	-	-
-	1	1	1	0	2	0	1	1	1	-
-	0	1	1	1	2	1	1	1	0	-
-	1	1	1	2	2	2	1	1	1	-
-	1	1	1	3	4	3	1	1	1	-
	1	1	3	3	13	3	3	1	1	-
	1	1	1	3	4	3	1	1	1	-
-	1	1	1	2	2	2	1	1	1	-
-	0	1	1	1	2	1	1	1	0	-
-	1	1	1	0	2	0	1	1	1	-
-	-	-	-	-	-	-	-	-	-	-

Figure 2(a-b): (a) each straight line is a (continuous) feature point spread function (fpsf) showing the effect (votes) in space B of the features in space A encoded as the associated integers. (b) In a digital implementation, thirteen 9-cell fpsfs are superimposed to give a parameter point spread function (ppsf)--the image in parameter space of a 13-feature instance. The peak corresponding to the instance is in the center.

References

Ballard, D.H., "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition 13*, 2, 111-122, 1981.

Brown, C.M., "Bias and noise in the Hough transform I: Theory," TR 105, Computer Science Dept., U. Rochester, June 1982; submitted to *IEEE Trans. on Pattern Analysis and Machine Intelligence.*

Brown, C.M. and M. Curtiss, "Bias and noise in the Hough transform II: Experiments," TR113, Computer Science Dept., U. Rochester, August 1982.

Brown, C.M. and D. Sher, "Modeling the sequential behavior of Hough transform schemes," TR114, Computer Science Dept., U. Rochester, 1982.

Duda, R.O. and P.E. Hart, "Use of the Hough transform to detect lines and curves in pictures," *CACM 15*, 1, 11-15, 1972.

Hough, P.V.C., *Method and Means for Recognizing Complex Patterns*, U.S. Patent 3069654, 1962.

O'Rourke, J., "Dynamically quantized spaces for focusing the Hough transform," *Proc.*, 7th IJCAI, 737-739, Vancouver, B.C., August 1981.

Shapiro, S.D., "Feature space transforms for curve detection," *Pattern Recognition 10*, 129-143, 1978a.

Shapiro, S.D., "Generalization of the Hough Transform for curve detection in noisy pictures," *Proc.*, 4th Int'l. Joint Conf. on Pattern Recognition, 710-714, Kyoto, Japan, November 1978b.

Shapiro, S.D., "Transformations for the computer detection of curves in noisy pictures," *Computer Graphics and Image Processing 4*, 328-338, 1975.

Sloan, K.R., Jr., "Dynamically quantized pyramids," *Proc.*, 7th IJCAI, Vancouver, B.C., August 1981.

```
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  1  - 10  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  2  - 11  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  3  - 12  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  4  - 13  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  5  - 14  -  -  -  -  1  - 10  -  -  -  -  -  -  -
 -  -  -  6  - 15  -  -  -  -  2  - 11  -  -  -  -  -  -  -
 -  -  -  7  - 16  -  -  -  -  3  - 12  -  -  -  -  -  -  -
 -  -  -  8  - 17  -  -  -  -  4  - 13  -  -  -  -  -  -  -
 -  -  -  9  - 18  -  -  -  -  5  - 14  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  6  - 15  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  7  - 16  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  8  - 17  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  9  - 18  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
                            (a)
```

```
 2  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 12  -  -  -  -
 -  -  -  8  -  -  -  -  - 15  -  -  -  -  -  -  -  7  -  -
 -  -  -  1  5 10  -  -  -  -  -  -  -  -  -  9  -  -  -  -
 -  -  -  2 12 11  -  - 16  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  3  - 12  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  4  - 13  9 10  -  -  -  -  -  4  - 14  -  -  4  -
 -  -  -  5  - 14  -  -  -  -  1  - 10  -  -  -  -  -  -  -
 -  -  -  6 12 15  -  -  -  - 15  - 11  - 11  - 17  -  -  -
 -  -  -  7  7 16  -  -  -  -  3  2 12  -  9  -  - 13  -  -
 - 11  2  8  - 17  -  -  -  -  4 18 13  4 12  1  -  7  -  -
 2  -  -  9  6 18  -  -  -  -  5  - 14  -  -  2  -  -  -  -
 -  -  -  -  -  -  -  -  -  5  6  - 15  -  -  -  -  -  -  -
 - 14  -  -  -  -  -  -  -  -  7  - 16  2  -  -  -  2  -  -
 -  -  -  -  - 11  -  -  -  -  8  -  1  -  - 16  -  -  -  -
 - 18  -  -  -  -  -  -  -  -  9  - 18  -  - 13  -  -  7  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  6 14  -  -
 -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
 -  -  -  -  -  -  -  -  -  -  -  6  -  -  -  -  -  -  -  -
 -  -  -  -  -  - 15  -  -  -  -  -  -  - 15  -  -  -  -  -
                            (b)
```

Figure 3(a-b): (a) Two 18-feature, two-part instances in a 20x20 abstract image. (b) The instances of (a) with 15% signal-independent noise.

```
Cache Length: 64.  Part length: 11.

Number of parts: 1

Noise %     0%                  |    15%                |    30%                |
Fpsf Len:   3      9      15    |    3      9      15   |    3      9      15   |

Avg:l)    2.75   2.75   2.75    |  2.38   1.52   1.29   |  1.88   0.35   0.50   |
StDev:    0.00   0.00   0.00    |  0.50   0.65   0.77   |  0.46   0.56   0.58   |
N:          10     10     10    |    10     10     10   |    10     10     10   |

Avg:u)    2.75   2.75   2.75    |  2.30   1.82   1.61   |  1.75   1.01   0.89   |
StDev.    0.00   0.00   0.00    |  0.48   0.32   0.35   |  0.28   0.33   0.51   |
N:          10     10     10    |    10     10     10   |    10     10     10   |

Avg:p)    2.75   2.75   3.03    |  2.47   1.78   1.41   |  1.88   0.85   0.89   |
StDev:    0.00   0.00   0.42    |  0.56   0.45   0.65   |  0.52   0.52   0.82   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Avg:r)    2.37   2.22   2.35    |  1.58   1.20   0.80   |  1.11   0.55   0.85   |
StDev:    0.61   0.47   0.52    |  0.51   0.48   0.58   |  0.61   0.51   0.53   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Avg:lp)   2.75   2.75   3.01    |  2.44   1.87   1.08   |  1.87   0.55   0.72   |
StDev:    0.00   0.00   0.41    |  0.55   0.65   0.84   |  0.55   0.66   0.89   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Number of parts: 2

Noise %     0%                  |    15%                |    30%                |
Fpsf Len:   3      9      15    |    3      9      15   |    3      9      15   |

Avg:l)    3.14   3.14   3.14    |  2.75   2.53   2.39   |  2.21   2.19   1.88   |
StDev:    0.00   0.00   0.00    |  0.30   0.32   0.46   |  0.32   0.48   0.46   |
N:          10     10     10    |    10     10     10   |    10     10     10   |

Avg:u)    3.14   3.14   3.14    |  2.70   2.53   2.51   |  2.38   1.96   1.67   |
StDev:    0.00   0.00   0.00    |  0.25   0.27   0.40   |  0.42   0.31   0.64   |
N:          10     10     10    |    10     10     10   |    10     10     10   |

Avg:p)    3.14   3.14   3.19    |  2.81   2.65   2.40   |  2.47   2.03   1.64   |
StDev:    0.00   0.00   0.17    |  0.35   0.46   0.58   |  0.59   0.47   0.57   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Avg:r)    2.55   2.63   2.86    |  2.31   2.15   1.85   |  1.82   1.43   1.17   |
StDev:    0.40   0.42   0.49    |  0.43   0.50   0.51   |  0.52   0.56   0.59   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Avg:lp)   3.14   3.14   3.22    |  2.80   2.82   2.56   |  2.39   2.15   1.74   |
StDev:    0.00   0.00   0.21    |  0.36   0.55   0.68   |  0.48   0.45   0.55   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Number of parts: 3

Noise %     0%                  |    15%                |    30%                |
Fpsf Len:   3      9      15    |    3      9      15   |    3      9      15   |

Avg:l)    3.30   3.30   3.30    |  2.94   2.75   2.98   |  2.86   2.66   2.35   |
StDev:    0.00   0.00   0.00    |  0.18   0.20   0.46   |  0.31   0.37   0.48   |
N:          10     10     10    |    10     10     10   |    10     10     10   |

Avg:u)    3.30   3.30   3.30    |  2.91   2.88   2.87   |  2.74   2.43   1.12   |
StDev:    0.00   0.00   0.00    |  0.18   0.30   0.47   |  0.28   0.17   1.15   |
N:          10     10     10    |    10     10     10   |    10     10     10   |

Avg:p)    3.30   3.31   3.42    |  2.85   2.77   2.71   |  2.86   2.87   2.27   |
StDev:    0.00   0.05   0.18    |  0.21   0.34   0.42   |  0.34   0.36   0.49   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Avg:r)    2.83   2.84   2.88    |  2.54   2.27   2.28   |  2.29   2.10   1.77   |
StDev:    0.40   0.37   0.44    |  0.43   0.46   0.54   |  0.47   0.48   0.65   |
N:         100    100    100    |   100    100    100   |   100    100    100   |

Avg:lp)   3.30   3.30   3.46    |  2.95   2.91   2.87   |  2.84   2.80   2.35   |
StDev:    0.00   0.00   0.29    |  0.23   0.49   0.62   |  0.42   0.46   0.39   |
N:         100    100    100    |   100    100    100   |   100    100    100   |
```

TABLE 2

Tables 1, 2, 3, 4: (Only Table 2 is included in DARPA-IU proceedings). Table rows are scanning methods: l) Left to Right within Top to Bottom. u) Top to Bottom within Left to Right. p) Random without replacement. r) Random with replacement. lp) Left to Right within permuted scanlines. Table entries are mean and standard deviation of peak/background ratio (eq. 1).

```
Cache Length: 32.  Scan: Random.  Noise: 15%

Fpsf length: 3

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4 ••
1.6 •••••••
1.8 ••••
2.0 ••••••••••••
2.2 •••••••••••
2.4 •••
2.6 ••••••••••••••••••
2.8 •••••••••
3.0 ••••••••••••••
3.2
3.4 ••••
3.6 •••
3.8
4.0 •••

Fpsf length: 9

0.0 •••••••••••
0.2 •
0.4
0.6 •
0.8 •••••••••
1.0 ••••••
1.2 ••••••
1.4 •••••••
1.6 •••••••••••••
1.8 ••••••••••
2.0 ••••••••
2.2 ••••••••••••
2.4 •••
2.6 •••••
2.8 ••
3.0 •
3.2
3.4 •
3.6 ••••
3.8
4.0

Fpsf length: 15

0.0 ••••••••••••••••••••••••••••• •••••••••••
0.2 ••••
0.4 •
0.6 ••
0.8 •
1.0 ••
1.2 ••••••
1.4 ••••••••
1.6 •••••••••
1.8 ••••••
2.0 ••
2.2 •••••
2.4 ••••
2.6 •
2.8 •
3.0 ••
3.2
3.4
3.6
3.8 •
4.0
```

```
Cache length: 64.  Scan: Random.  Noise: 15%

Fpsf length: 3

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6 •••••••••••••
1.8
2.0 ••••••••••
2.2 ••••••••••••
2.4 •
2.6 ••••••••••••••••
2.8 •••••••••
3.0 •••••••••••••••••
3.2
3.4 •••
3.6 ••
3.8
4.0

Fpsf length: 9

0.0
0.2
0.4
0.6
0.8 ••••
1.0 ••••
1.2 ••••••
1.4 •••••••••••••••
1.6 •••••••••••
1.8 •••••••••••••••••••••
2.0 ••••••••••••
2.2 •••••••••
2.4 ••••••
2.6 ••••
2.8 •••
3.0
3.2
3.4
3.6
3.8
4.0

Fpsf length: 15

0.0 •••••••
0.2 •••
0.4 •
0.6 ••••
0.8 ••
1.0 ••••••
1.2 •••••••••••••
1.4 •••••••••
1.6 •••••••••••••••••••
1.8 ••••••••••
2.0 ••••••
2.2 •••••••
2.4 •••••
2.6 ••
2.8
3.0
3.2
3.4
3.6
3.8
4.0
```

Figure 4(a-p): (Only (e) and (f) are included in DARPA-IU proceedings). Histograms of peak/background ratios from a subset of cases in Tables 1-4.

```
Cache length: 32. Scan: ItoB, ItoR.
Noise: 15%.  Instances: 1.
Parts: 1.  Fpsf length: 9.
Part length: 11.  N: 100

Flush: Threshold

Mean of Ratios:         1.212250
Std. Dev. of Ratios: 0.972063

0.0 •••••••••••••••••••••••••••••
0.2
0.4
0.6 ••
0.8 •••
1.0 •••••••
1.2 ••••••••••••
1.4 ••••
1.6 •••••••••
1.8 •••••••
2.0 •••••••
2.2 •••••••
2.4 ••
2.6 •••••
2.8
3.0 •••••
3.2
3.4
3.6 •
3.8
4.0

Flush: Random Below Threshold

Mean of Ratios:         1.578250
Std. Dev. of Ratios: 0.895189

0.0 •••••••••••••••
0.2
0.4 ••
0.6 •••
0.8 •
1.0 ••••
1.2 ••••••
1.4 •••••••
1.6 •••••••••••
1.8 •••••••••••
2.0 •••••••••
2.2 •••••••••••
2.4 •••
2.6 ••••••••••
2.8 ••
3.0 •
3.2 ••
3.4 ••
3.6
3.8
4.0
```

```
Cache Length: 64. Scan: ItoB, ItoR.
Noise: 15%.  Instances: 1.
Parts: 1.  Fpsf length: 9.
Part Length: 11.  N: 100

Flush: Threshold

Mean of Ratios:          1.805774
Std. Dev. of Ratios:  0.501593

0.0 •
0.2
0.4
0.6
0.8
1.0 ••
1.2 •••••••••••
1.4 ••••••••••••
1.6 •••••••••••••
1.8 •••••••••••••••••••••••••
2.0 •••••••••••
2.2 •••••••••
2.4 •••••••
2.6 ••••••
2.8
3.0 •
3.2 •
3.4 •
3.6
3.8
4.0

Flush: Random Below Threshold

Mean of Ratios:         1.791405
Std. Dev. of Ratios: 0.461134

0.0 •
0.2
0.4
0.6
0.8 •
1.0 •••
1.2 ••••••••
1.4 ••••••••••
1.6 •••••••••••••••
1.8 ••••••••••••••••••••••••
2.0 ••••••••••••••••
2.2 ••••••••••••
2.4 •••
2.6 •••••
2.8
3.0 •
3.2 •
3.4
3.6
3.8
4.0
```

Figure 5(a-d): (Only (a) and (b) are included in DARPA-IU proceedings). Comparison of two Flush algorithms. Histograms as in Fig. 4.

CONTOUR-BASED MOTION ESTIMATION

AD-P000 119

Larry S. Davis
Zhongquan Wu
Hanfang Sun

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

This paper introduces a contour-based approach to motion estimation. It is based on first computing motion at image corners, and then propagating the corner motion estimates along the principal contours in the image based on a local 2½D motion assumption. The results of several experiments are presented.

1. Introduction

The earliest problem that arises in the analysis of time-varying images is the detection of moving image elements (edge, regions) and the computation of the image velocity (optic flow) of those elements. A variety of computational schemes have been proposed to solve this problem. In a recent survey, Ullman [1] broadly classifies these as intensity-based and token-matching schemes.

An important class of intensity-based schemes takes advantage of the relationship between the temporal and spatial gradient of any continuous and differentiable image property which is invariant to small changes in perspective. For example, if we assume that the intensity, I, satisfies these properties, the relationship

$$-I_t = uI_x + vI_y \qquad (1)$$

can be used to determine velocity. Here I_t is the temporal intensity gradient, I_x and I_y the x and y components of the spatial intensity gradient, and u and v the x and y components of image velocity. Measuring I_t, I_x, I_y from an image sequence establishes a linear constraint on the x and y velocity components. A single velocity estimate can be computed by spatially combining the constraints using e.g., Hough transforms [2], least-squares methods [3] or minimization techniques [4]. All of these techniques suffer from certain disadvantages. The Hough-transform and minimization techniques assume that image velocity is uniform over large parts of the image, and the least-squares method further assumes that the constraint equations determined for nearby points are independent - as assumption that is violated by the spatial integration required to compute spatial derivatives.

In this paper we develop a contour-based approach to motion estimation at a small set of image points at which it is possible, in principle, to unambiguously determine image velocity. Specifically, corners have the property that their motion can be directly computed based only on measurements made at the corner (in practice, of course, one must examine a small neighborhood of the corner). Another important property of corners is that they can be safely regarded as projections of scene features whose general appearance is invariant to rigid motion - e.g., an image corner may be the projection of the vertex of a polyhedron, or of a curvature discontinuity on the boundary of a surface marking. The second step involves propagating these velocity estimates to a larger number of picture points. This is based on the assumption that the image motion is locally a rigid-two-dimensional motion. Given this assumption, the velocity at one point on a contour and the normal component at a neighboring point can be combined to compute the actual velocity at the neighboring point.

2. Estimating motion at corners

The motion of a corner can be computed in a variety of ways. Section 2.1 describes an approach based on temporal intensity changes along lines parallel to the sides of the corner. Section 2.2 discusses a second technique which combines normal vectors in a small neighborhood of the corner along the contour. It is similar to the velocity estimation algorithm in Horn and Schunck [4]. We should also point out that Nagel [5] has recently proposed a corner velocity estimation algorithm based on a differential approach (i.e., a Taylor series expansion of the image function in the neighborhood of the corner truncated after the second-order terms.).

2.1 A structural approach

This subsection presents a structural approach to corner motion estimation. We first describe velocity computation for the case of translation motion, and then consider translation combined with rotation.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Clara Robertson in preparing this paper. This paper is based in part on two earlier reports (TR's 1130 and 1132).

Suppose that a corner simply translates from point C_0 to C_1 between two frames t_0 and t_1 (see Figure 1). Let O' be a point on the bisector of PC_0R and let O'A and O'B be lines parallel to C_0R, respectively at some unit distance from C_0P and C_0R. Suppose that $|O'A| = |O'B| = 1+m$, for some constant m. Finally, assume that the intensity inside the corner is 1 and outside the corner is 0.

Now, at time t_0, the average intensity along line segments O'A and O'B is

$$I_{O'A}(t_0) = I_{O'B}(t_0) = 1/(1+m)$$

If $\Delta x'$ and $\Delta y'$ are the components of the translation in the directions of the lines O'A and O'B, then

$$I_{O'A}(t_1) = (1+\Delta x')/(1+m)$$

$$I_{O'B}(t_1) = (1+\Delta y')/(1+m)$$

assuming that m is chosen large enough so that $\max(\Delta x', \Delta y') < m$. Finally, $\Delta x'$ and $\Delta y'$ can be computed from

$$I_{O'A} = I_{O'A}(t_1) - I_{O'A}(t_0) = \Delta x'/(1+m)$$

$$I_{O'B} = I_{O'B}(t_1) - I_{O'B}(t_0) = \Delta y'/(1+m)$$

Once $\Delta x'$ and $\Delta y'$ are computed, the components of the velocity in the original image coordinate system can be recovered easily:

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \cos\beta \\ \sin\alpha & \sin\beta \end{bmatrix} \begin{bmatrix} \Delta x' \\ \Delta y' \end{bmatrix}$$

The practical success of this technique depends on our ability to compute several corner parameters accurately. These parameters are

1. corner location at t_0,
2. corner shape (angles α and β), and
3. corner contrast (assumed here to be 1)

The computation of these parameters is discussed in Section 4.1. Next, we extend the previous simple analysis to include rotation as well as translation. We will treat this case as a translation from C_0 to C_1 followed by a rotation about C_1 through a clockwise angle γ (see Figure 2). Since translation and rotation are specified by a total of three parameters, we could extend the above analysis using only a third line segment parallel to either O'A or O'B. Instead, we consider two pairs of parallel line segments, and compute the displacements in the directions O'A and O'B rather than directly computing the angle γ.

Let $\Delta x'_t$, $\Delta y'_t$ be the translational components of the motion in the O'A and O'B directions, and $\Delta x'_r$ and $\Delta y'_r$ the corresponding rotational components. Then

$$\Delta x'_t + \Delta x'_r = (1+m)\, \Delta I_{O'A} \qquad (2.1)$$

$$\Delta y'_t - \Delta y'_r = (1+m)\, \Delta I_{O'B} \qquad (2.2)$$

From Figure 3, we see that

$$\frac{\Delta x'_r}{1+\Delta y'_t} = \frac{\Delta x'_r - \Delta x''_r}{\delta} \qquad (2.3)$$

where δ is the distance between the parallel line segments O'A and CD. Similarly

$$\frac{\Delta y'_r}{1+\Delta x'_t} = \frac{\Delta y'_r - \Delta y''_r}{\delta} \qquad (2.4)$$

Also

$$\Delta x'_r - \Delta x''_r = (1+m)\,[I_{O'A}(t_1) - I_{CD}(t_1)] \qquad (2.5)$$

$$\Delta y'_r - \Delta y''_r = (1+m)\,[I_{O'B}(t_1) - I_{EF}(t_1)] \qquad (2.6)$$

Substituting (2.5) and (2.6) into (2.3) and (2.4) and simplifying, we obtain

$$\Delta x'_r - C_1 \Delta y'_t = C_1 \qquad (2.7)$$

$$C_2\, \Delta x'_t - \Delta y'_r = -C_2 \qquad (2.8)$$

where

$$C_1 = \frac{(1+m)}{\delta}\,[I_{O'A}(t_1) - I_{CD}(t_1)]$$

$$C_2 = \frac{(1+m)}{\delta}\,[I_{EF}(t_1) - I_{O'B}(t_1)]$$

Solving for $\Delta x'_t$ and $\Delta y'_t$ we obtain

$$\Delta x'_t = \frac{(1+m)\,(\Delta I_{O'A} - C_1 \Delta I_{O'B}) - C_1(1+C_2)}{1+C_1C_2} \qquad (2.9)$$

$$\Delta y'_t = \frac{(1+m)\,(C_2 \Delta I_{O'A} - C_1 \Delta I_{O'B}) - C_2(C_1-1)}{1+C_1C_2} \qquad (2.10)$$

Substituting (2.7) and (2.8) into (2.9) and (2.10), we can also compute $\Delta x'_r$ and $\Delta y'_r$, which gives us a complete description of the motion of the corner.

2.2 A least-squares approach

In this section we show how simple least-squares algorithms can be used to compute corner motion. We can rewrite (1.1) to obtain

$$V_n = I_t / |\nabla I| \qquad (2.11)$$

where $|\nabla|$ is the magnitude of the spatial intensity gradient at that point ($|\nabla I| = \sqrt{I_x^2 + I_y^2}$ and V_n is the projection of the velocity vector $\underline{V}$ onto the intensity gradient at that point. We will first consider the case when the velocity is only a translation, and then consider translation with rotation.

If we assume that the velocities are constant in a small neighborhood of the corner along the contour, then we can relate the problem of determining the velocity $\underline{V}$ at the corner to that of determining a $\underline{V}$ to minimize

$$E = \sqrt{\{\Sigma\,(\underline{V}\cdot\bar{n}_i - V_{ni})^2\}} \qquad (2.12)$$

where $\bar{n}_i$ (n_{i1}, n_{i2}) is the unit normal vector of the ith contour point and V_{ni} is the projection of $\underline{V}$ onto the intensity gradient direction at the ith contour point, called the normal projection for short. By minimizing the error E^2, we obtain

$$\begin{aligned} au + cv &= d \\ cu + bv &= e \end{aligned} \qquad (2.13)$$

where

$$a = \sum_{i=1}^{k} n_{i1}^2$$

$$b = \sum_{i=1}^{k} n_{i2}^2$$

$$c = \sum_{i=1}^{k} n_{i1}\cdot n_{i2}$$

$$d = \sum_{i=1}^{k} n_{i1}\cdot V_{ni}$$

$$e = \sum_{i=1}^{k} n_{i2}\cdot V_{ni}$$

From equation (2.12), we have the velocity estimation of a turning point

$$u = \frac{bd-ce}{ab-c^2}, \qquad v = \frac{ae-cd}{ab-c^2} \qquad (2.14)$$

Notice that the solutions for u and v are only meaningful when $ab-c^2$, which is related to the variance of normal directions, is high. For a straight line segment, e.g., there is no solution because the denominators of eqs. (2.13) are zero,

$$\begin{aligned} ab-c^2 &= \sum_{i=1}^{k} n_{i1}^2 \cdot \sum_{i=1}^{k} n_{i2}^2 - (\sum_{i=1}^{k} n_{i1}\cdot n_{i2})^2 \\ &= kn_1^2\cdot kn_2^2 - (kn_1\cdot n_2)^2 \\ &= 0 \end{aligned}$$

i.e., from a small element of a straight line, the only information that one can obtain is the motion component normal to that line, and motion along this line element cannot be detected. Corners, however, are just those points where the variance of normal directions is locally maximal.

Now consider the case in which the motion of the contour can be decomposed into a translation with velocity $\underline{V}_0$ (V_{0x}, V_{0y}) at the corner (x_0, y_0) and a rotation around (x_0, y_0) with the angular velocity $\underline{w}$, as shown in Figure 4. Note that since we are only interested in the motion of the corner, we do not explicitly solve for w, although it would be easy to do so.

We obtain
$$\underline{w} \times \underline{d}_i = \underline{V}_i - \underline{V}_0 \qquad (2.15)$$

By rewriting this equation, we have

$$\underline{V}_i = \begin{pmatrix} V_{0x} & -wd_{iy} \\ V_{0y} & +wd_{ix} \end{pmatrix} \qquad (2.16)$$

where d_{iy}, d_{ix} are the components of the displacement vector d_i from point (x_0, y_0) to point (x_i, y_i) on the contour. The normal component of $\underline{V}_i$ is related to $\underline{d}_i$, w and $\underline{V}_0$ by

$$\begin{aligned} V_{ni} &= \underline{V}_i\cdot\bar{n}_i \\ &= (V_{0x} - wd_{iy})n_{ix} + (V_{0y} + wd_{ix})n_{iy} \end{aligned} \qquad (2.17)$$

By considering three points on the contour, V_{0x} and V_{0y} can be simply obtained by solving linear equations. In general, more than three image points are taken and V_{0x}, V_{0y} are computed by minimizing the following square error:

$$E^2 = \sum_i (V_{0x}\cdot n_{ix} + V_{0y}\cdot n_{iy} + w(d_{ix}n_{iy} - d_{iy}n_{ix}) - V_{ni})^2$$

We obtain the following least squares solution of V_{0x}, V_{0y}:

$$V_{0x} = \frac{\begin{vmatrix} C_1 & S_{11} & a_1 \\ C_2 & S_{02} & a_2 \\ C_3 & S_{01} & a_3 \end{vmatrix}}{\delta} \qquad V_{0y} = \frac{\begin{vmatrix} S_{20} & C_1 & a_1 \\ S_{11} & C_2 & a_2 \\ S_{10} & C_3 & a_3 \end{vmatrix}}{\delta} \qquad (2.18)$$

where

$$S_{rpqk} = \sum_i n_{ix}^r\ n_{iy}^p\ d_{ix}^q\ d_{iy}^k$$

$$\delta_{rp} = \sum_i n_{ix}^r\ n_{iy}^p$$

and

$$a_1 = S_{1110} - S_{2001},$$

$$a_2 = S_{0210} - S_{1101},$$

$$a_3 = S_{0110} - S_{1001},$$

$$c_1 = \sum_i V_{xi}\cdot n_{ix},$$

$$c_2 = \sum_i V_{ni}\cdot n_{iy},$$

$$c_3 = \sum_i V_{ni},$$

and

$$S = \begin{vmatrix} S_{20} & S_{11} & a_1 \\ S_{11} & S_{02} & a_2 \\ S_{10} & S_{01} & a_3 \end{vmatrix}$$

3. Propagation of velocity vectors along image contours

3.1 The local constraint and the propagation formula

Suppose the velocity vectors $\underline{V_0}, \underline{V_k}$ at the ends of a contour A_0A_k are known (see Figure 4). Consider a small line segment dS along the contour A_0A_1. Assuming that the motion is a rigid motion V_{0S} of V_0, the motion of A_0 parallel to A_0A_1 must equal the parallel component V_{1S} of the velocity V_1 at A_1:

$$V_{0S} = V_{1S} \tag{3.1a}$$

or

$$\underline{V_0} \cdot \overline{dS} = \underline{V_1} \cdot \overline{dS}. \tag{3.1b}$$

where $\underline{V_0}$ and V_1 are the velocity vectors at the two ends of the line dS, and $\overline{dS}$ is the unit vector along dS, the vector joining A_0 to A_1. Rewriting this local constraint (eq. 3.1b) into component form, we obtain

$$\underline{V_0} \cdot \overline{dS} = (V_{1n}\bar{n} + V_{1t}\bar{t}) \cdot \overline{dS}$$

$$= V_{1n}\bar{n} \cdot \overline{dS} + V_{1t}\bar{t} \cdot \overline{dS} \tag{3.2}$$

where V_{1n} and V_{1t} are the normal component and the tangential component of the velocity vector V_1 respectively, and $\bar{n}$ and $\bar{t}$ are the unit vectors in the normal and tangent directions of the contour at A_1. From Figure 5, we see that

$$\bar{n}.\overline{dS} = \cos\alpha \text{ and } \bar{t}.\overline{dS} = \cos(\pi/2-\alpha) = \sin\alpha$$

so that

$$V_{0S} = V_{1t}\sin\alpha + V_{1n}\cos\alpha \tag{3.3}$$

Thus, we have that the tangential component is

$$V_{1t} = (V_{0S} - V_{1n} \cdot \cos\alpha)/\sin\alpha \tag{3.4}$$

where α is the angle between the unit vector $\overline{dS}$ and the normal vector $\bar{n}$ at the point A_1. We also have $\gamma=\beta-\alpha$, where β is the angle between the x-axis and the normal vector $\bar{n}$, and γ is the angle between the x-axis and the line segment dS.

We can propagate the velocity along a contour using eq. (3.4), because the first projection V_{0S} is known after the previous propagation and the normal component V_{1n} can be computed by, e.g., the methods discussed in [3] or [4]. Once V_{1n} is computed, $V_1=V_1e^{j\theta}$ can be obtained because

$$V_1 = \sqrt{V_{1n}^2+V_{1t}^2} \tag{3.5}$$
$$\theta = \beta - \arctan V_{1t}/V_{1n}$$

3.2 Error analysis and a correction technique

From eq. (3.4) the new estimate of the tangent component V_{1t} is based on the previous projection V_{0S} and on the normal component V_{1n} at the current propagation point. Differentiating this equation we obtain

$$dV_{1t} = \frac{\partial V_{1t}}{\partial V_{0S}} dV_{0S} + \frac{\partial V_{1t}}{\partial V_{1n}} dV_{1n} + \frac{\partial V_{1t}}{\partial \alpha} d\alpha$$

$$= \frac{1}{\sin\alpha} dV_{0S} - \cot\alpha\, dV_{1n} + \frac{(V_{1n}-V_{0S}\cos\alpha)}{\sin^2\alpha} d\alpha \tag{3.6}$$

Note that the error in V_{1t} depends on the error in the previous projection (dV_{0S}), the error in the normal component V_{1n} at the current propagation point (dV_{1n}), and the error in the measurement of the angle α ($d\alpha$).

The result of these various errors is that when the propagation reaches A_k, the velocity vector V_k' attributed to A_k by the propagation procedure will differ from the velocity vector originally computed at A_k. Therefore, at the point A_k we compute the error between the propagation velocity estimate V_k' and the original velocity vector $\underline{V_k}$ and compute the error

$$\Delta\underline{V_k} = \underline{V_k} - V_k'$$

If this error is less than some tolerance, then this propagation procedure is stopped at point A_k; otherwise a correction procedure is applied. If we consider the error ΔV_k as having been accumulated in the previous n steps, then the average velocity error in one step is

$$\underline{V_e} = \Delta\underline{V_k}/k$$

so we have $m.V_e$ as the velocity error at the m^{th} step and we propagate this velocity error step by step backward to correct the estimated velocity vector at each point along the same contour.

4. Experimental results

We applied the corner motion estimation and velocity propagation algorithms to two sets of motion pictures. Section 4.1 describes the corner motion estimation results, and Section 4.2 describes the propagation results.

4.1 Corner motion estimation

The three corner motion models described in Section 2 were applied to two image sequences containing two frames each (Figures 6-7).

We first describe the application of the structural model presented in Section 2.1. Corners are "provisionally" detected using the corner detection algorithm described in Kitchen and Rosenfeld [6]. Next, a small window around each corner is analyzed to obtain a more accurate description of the corner. Based on the assumption that the corner locally contrasts with its surround, a local thresholding procedure (Milgram [7]) is used to segment the window. The corner is then relocated to a maximum curvature boundary point in the thresholded window. The slopes of the line segments meeting at the corner are computed using a one-dimensional (slope) Hough transform procedure (only slope need be computed since the lines are constrained to pass through the corner point.) The corners detected by this procedure are marked with dark crosses in Figures 6a and 7a.

To overcome the effects of various sources of error on the motion estimation, several quadruples of line segments are used to compute estimates of $\Delta x'_t$, $\Delta y'_t$, $\Delta x'_r$, and $\Delta y'_r$, with the final motion estimate taken as the average.

The results for the airplane in Figure 6 are displayed in Table 1. The estimated motion vectors were obtained by the authors' examination of digital enlargements of the images. No useful results were obtained for the moving car in Figure 7. There are several reasons for this:

1. The grey level corners in the car are much more rounded than the airplane's, and the motion estimates are sensitive to the corner location; and

2. The spatial resolution of Figure 7 is not high enough to allow us to place a sufficiently large window around a corner for segmentation which does not contain some other image feature.

The least-squares corner motion estimates presented in Section 2.2 require that we first compute the normal component of motion along the contour in the neighborhood of the corner. The magnitude of the normal component and the components of the unit normal vector on the x and y axes are

$$V_n = -I_t / \sqrt{I_x^2+I_y^2} \quad (4.1)$$

$$n_1 = I_x / \sqrt{I_x^2+I_y^2} \quad (4.2)$$

$$n_2 = I_y / \sqrt{I_x^2+I_y^2} \quad (4.3)$$

The derivatives (I_x, I_y, I_t) in (4.1)-(4.3) are approximated as follows:

$$I_x \doteq \frac{1}{4} \sum_{\ell=0}^{1} \sum_{m=0}^{1} \{I_{k+\ell,i+m,j+1} - I_{k+\ell,i+m,j}\}$$

$$I_y \doteq \frac{1}{4} \sum_{\ell=0}^{1} \sum_{n=0}^{1} \{I_{k+\ell,i+1,j+n} - I_{k+\ell,i,j+n}\}$$

$$I_t \doteq \frac{1}{4} \sum_{m=0}^{1} \sum_{n=0}^{1} \{I_{k+1,i+m,j+n} - I_{k,i+m,j+n}\}$$

The unit of length is the grid spacing interval in each image frame and the unit of time is the image frame sampling period.

Tables 2 and 3 contain the results of applying both least-squares corner estimators to the images in Figures 6 and 7, respectively.

4.2 Velocity propagation

We applied the propagation technique to the two image sequences displayed in Figures 6 and 7.

The propagation technique was implemented as follows:

1) Velocity vectors are first determined at a set of "corner" points in the first frame by the least-square corner motion estimator which assumes that the corner motion is a 2-D translation.

2) The velocity vector at the corner is propagated along the contours that meet at the corner until a second corner point is encountered. The contours are followed by a very simple maximum gradient technique. A velocity vector is not computed at every pixel on the contour, but only at every kth pixel, to reduce the error in α.

3) When the terminating corner point is reached, the propagation is stopped and the error velocity vector is computed. If this error is greater than a preset tolerance, then the error velocity vector is back-propagated along the same contour.

The results of the propagation are displayed in Figures 8-9.

5. Summary

We have presented a contour-based approach to motion estimation based on first estimating motion at image corners and then propagating these motion estimates along image contours. One potential advantage of such an approach over others such as [3-4] is that motion information is not integrated across the boundaries of moving objects, but only along such boundaries. Since very often the only reliable source of motion information is at object boundaries (when, for example, object interiors are homogeneous) it is important that motion estimation techniques yield accurate motion estimates at boundaries.

The examples presented in Section 4 both consisted of a single object moving across a homogeneous background. The propagation technique presented in Section 3 would need to be modified to be applicable to more complex image sequences containing multiple moving objects and occlusions so that motion information is not propagated from one object to another.

REFERENCES

1. Ullman, S., "Analysis of visual motion by biological and computer systems," Computer, 14, 8, 57-69, 1981.

2. Thompson, W. B. and S. T. Barnard, "Lower-level estimation and interpretation of visual motion," Computer, 14, 8, 20-28, 1981.

3. Glazer, F., "Computing optical flow," Proc. 7th Int. Joint Conf. Artificial Intelligence, Vancouver, B.C., 644-647, August 1981.

4. Horn, B. K. P. and B. G. Schunk, "Determining optical flow," A.I., 17, 1981, 185-204.

5. Nagel, H. H., "Displacement vectors derived from second order intensity variations in image sequences, Feb. 1982.

6. Kitchen, L. and A. Rosenfeld, "Gray level corner detection," University of Maryland Computer Science TR-887, April 1980.

7. Milgram, D., "Region extraction using convergent evidence," Comp. Graphics and Image Processing, 11, 1-12, 1979.

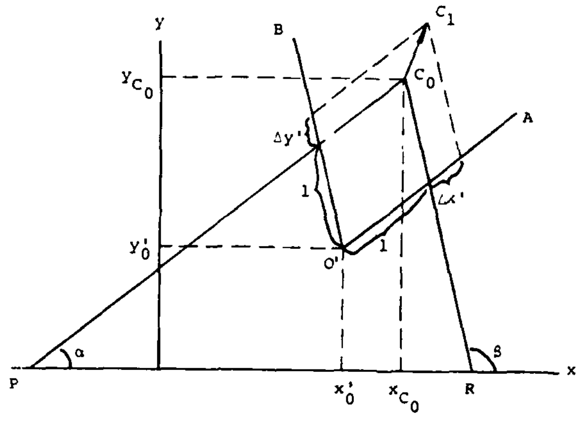


Figure 1

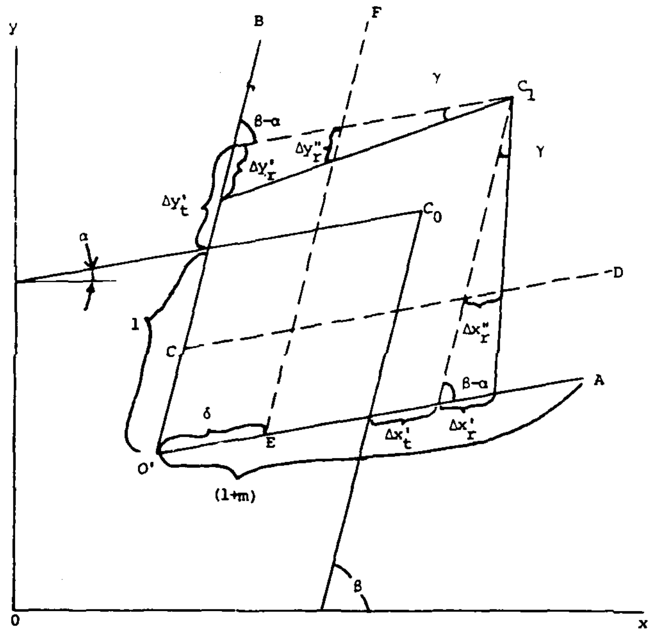


Figure 2

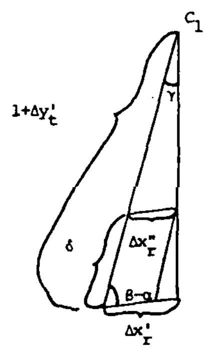


Figure 3

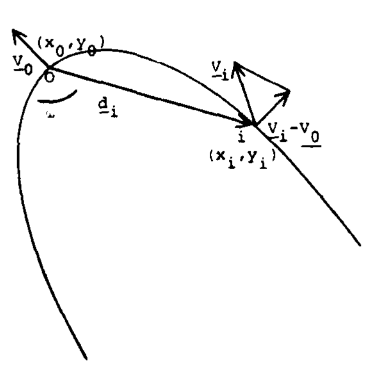


Fig. 4. Two-dimensional rigid motion.

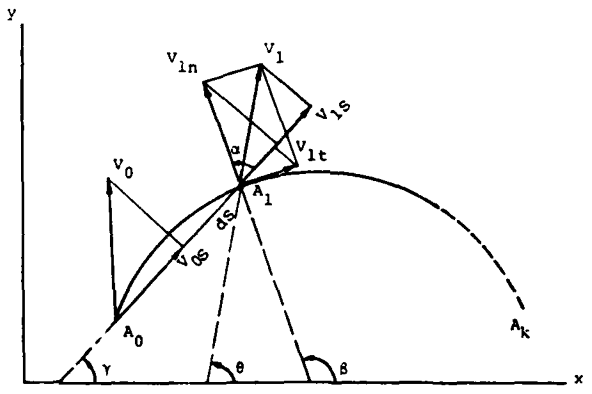
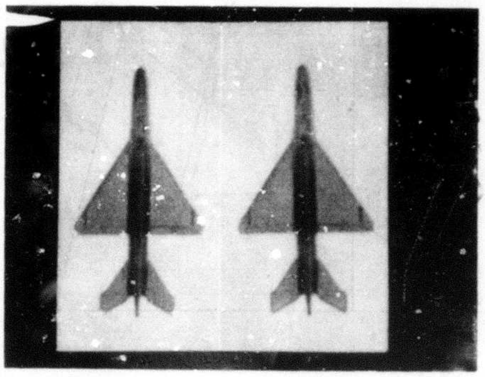
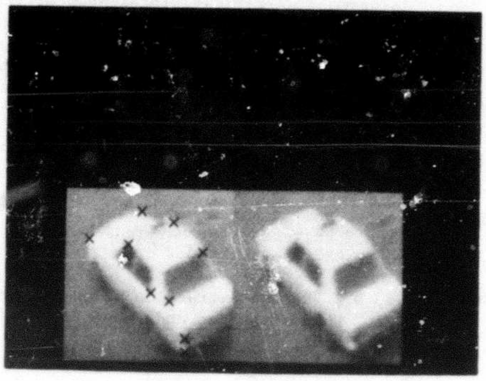


Figure 5. The geometry of the propagation along a contour in an image.



(a) (b)

Figure 6. Airplane sequence.



(a) (b)

Figure 7. Traffic sequence

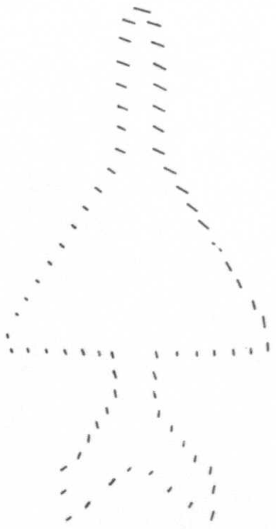


Figure 8. Velocity field using the propagation technique along the contours of the moving airplane shown in Figure 6.

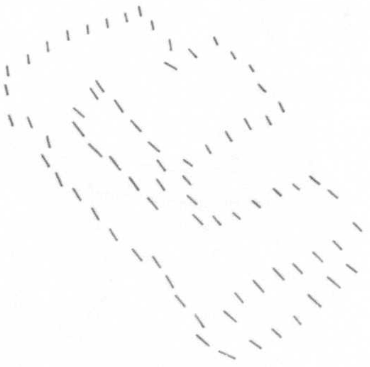


Figure 9. Velocity field using the propagation technique along the contours of the moving car shown in Figure 7.

	x	y	alpha	beta	measured u	measured v	estimated u	estimated v
1	39	93	170.1	213.5	-8.4	-1.5	-1.2	-5.9
2	166	46	264.5	345.6	-0.2	-0.8	-0.2	-1.0
3	166	146	16.4	96.6	-2.7	-1.8	-3.2	-1.5
4	167	87	89.5	185.4	-2.3	-1.6	-1.0	-0.9
5	168	104	175.1	276.6	-1.7	-1.2	-2.1	-0.6
6	212	123	39.2	195.3	-0.7	1.6	-2.7	0.6
7	227	68	9.6	292.6	-1.0	0.7	-1.0	1.3
8	227	124	70.2	353.8	-2.8	1.0	-2.9	1.0

Table 1. Motion vectors for corners in Figure 6.

x	y	estimated u	estimated v	computed by (2.14) u	computed by (2.14) v	computed by (2.18) u	computed by (2.18) v
39	93	-1.2	-5.9	3.1	-2.7	2.7	-0.5
166	46	-0.2	-1.0	-0.1	-0.8	0.1	-0.7
166	146	-3.2	-1.5	-1.2	-0.8	-0.2	-1.7
167	87	-1.0	-0.9	-0.9	-0.8	-1.0	-0.9
168	104	-2.1	-0.6	-1.8	-0.7	-2.4	-0.6
212	123	-2.7	-0.6	-1.7	1.0	-3.8	0.5
227	68	-1.0	1.3	-0.9	1.2	-1.0	0.6
227	124	-2.9	1.0	-2.4	0.1	-3.5	3.7

Table 2. Motion vectors for turning points of Figure 6 (airplane).

x	y	estimated u	estimated v	computed by eq. (2.14) u	computed by eq. (2.14) v	computed by eq. (2.18) u	computed by eq. (2.18) v
9	29	-1.5	-0.2	-1.4	-0.03	-1.4	-0.1
19	9	-1.5	-0.3	-1.3	-0.1	-1.4	-0.1
61	46	-1.8	-2.2	-1.1	-1.7	-1.8	-2.4
56	46	-1.6	-2.2	-1.5	-1.5	-0.8	-2.7
23	52	-1.6	-1.0	-1.0	-1.3	-1.0	-1.3
13	41	-1.2	-0.6	-1.2	-0.5	-1.2	-0.6
21	24	-1.8	-1.5	-1.4	-0.5	-1.6	-1.8
39	33	-1.8	-1.5	-1.7	-1.1	-1.4	-2.5
41	40	-1.3	-1.5	-2.1	-1.2	-0.4	3.0

Table 3. Motion vectors for turning points of Figure 7 (traffic).

AD-P000 120

THE RECOVERY OF SURFACE ORIENTATION FROM IMAGE IRRADIANCE.

Grahame Smith

Artificial Intelligence Center, SRI International
Menlo Park, California 94025

ABSTRACT

The image irradiance equation constrains the relationship between surface orientation in a scene and the irradiance of its image. Additional constraints, needed to recover surface orientation from image irradiance, usually require the recovered surface to be smooth. We demonstrate that smoothness is not sufficient for this task.

The 'beliefs' a visual system has about the laws of surface radiance provide sufficient constraints to relate the irradiance of an image to the corresponding scene's surface orientations. We propose a set of beliefs and derive surface orientation - image irradiance equations for a visual system with these beliefs. The surface orientations, derived from the image by a visual system with these beliefs, may not be those present in the scene, rather they are those believed to be present.

1 INTRODUCTION

Most previous work that has addressed the problem of the recovery of surface shape from shading (image irradiance) has been based on solving the image irradiance equation which relates the radiance of a scene to the irradiance of its image [1-8]. This formulation of the relationship between scene radiance and image irradiance is embodied in a first-order partial differential equation. The approaches to solving this differential equation have generally been either direct integration along characteristic curves [1], or an iterative algorithm that attempts to reduce the difference between the predicted image irradiance and the recorded value [5-7].

As the image irradiance equation is a single equation relating the image irradiance and two independent variables (specifying surface orientation), it does not uniquely determine the two independent variables for a given value of image irradiance. Consequently when this equation is used to recover surface shape additional constraints are necessary. These may be imposed by boundary conditions, by restrictions on the type of surface to be recovered, or by a combination of the two. For some images, when we can determine important features (such as the fact that an edge is an occlusion boundary caused by a surface turning smoothly away from the viewing direction), we can use boundary conditions to constrain the solution; in large portions of the image, however we can say something only about the type of surface we would like to recover. Surface smoothness is the weakest assumption to date that still allows surface shape to be recovered. Smoothness normally signifies that the surface is continuous and that it is once or twice differentiable. Smoothness has been required to play the role of propagator of boundary conditions and selector of the surface to be recovered. Is smoothness capable of these tasks?

Not all authors have used smoothness as their additional constraint. Pentland's additional constraints are concerned with surface shape [8]. He assumes that locally the surface has equal curvature along orthogonal directions. This is an assumption that is strong enough to allow but a single interpretation for the surface orientation and, at the same time, is one that enables recovery of the surface orientation by purely local computation.

In the new formulation presented here, we attempt to avoid assumptions about the particular form of $R(p,q)$ or of the surface shape. Rather we define a class of surface radiance functions that includes those functional forms generally used for surface radiance, and for this class we derive equations relating surface orientation to image irradiance.

The class of surface radiance functions is defined by two properties of the class. One may view these properties as axioms that a visual system uses to define its beliefs about the physical nature of the world. When our formulation is viewed in this manner, our assumptions are about the properties of reflection in the world rather than about specific reflection functions or specific surface shapes.

The presentation we adopt in this paper is to describe the various formulations that have been used to employ smoothness, including a relaxation procedure of our own that resembles its counterpart in engineering, and then to derive general surface orientation image irradiance equations. We look at some restricted situations that may prove useful in using this formulation to recover surface shape.

2 ITERATIVE FORMULATIONS FOR SURFACE RECOVERY

The image irradiance equation as presented by Horn [2] is

$$I(x,y) = R(p,q) \quad ,$$

where $I(x,y)$ is the image irradiance as a function of the image coordinates x and y, and $R(p,q)$ is the surface radiance as a function of p and q, the derivatives of depth with respect to the image coordinates. In the 'shape from shading' approach it is generally assumed that $R(p,q)$ is known for all p and q (that is, the reflectance map is specified). The iterative approach applies this equation on a pixel-by-pixel basis, that is for pixel (i,j)

$$I_{i,j} = R(p_{i,j}, q_{i,j}) \quad ,$$

where $I_{i,j}$ is the image irradiance for (i,j)th pixel, and $p_{i,j}$, $q_{i,j}$ is the surface orientation of the surface patch that is imaged at pixel (i,j). For convenience we use the notation

$$R_{i,j} \equiv R(p_{i,j}, q_{i,j}) \quad .$$

If, at some stage of the iterative procedure, we have assigned particular $p_{i,j}, q_{i,j}$ as the surface orientation of the (i,j)th pixel then the residual expression

$$\xi_{i,j}{}^R = (I_{i,j} - R_{i,j})^2$$

specifies the error caused by our assignment of surface orientation. If this were our only constraint we could select $p_{i,j}, q_{i,j}$ so that $\xi_{i,j}{}^R = 0$. This would guarantee that the image irradiance equation is satified pixel by pixel, but because there are infinitely many solutions, we need further constraints to reduce the number of possible solutions.

Smoothness is usually introduced by specifying a relationship that we would like to have hold between the surface orientation of the (i,j)th pixel and its neighbors. The various iterative approaches [5-7] differ in the way this relationship is specified. Of course, at a particular stage of the iterative process this relationship between neighboring pixels will not be exact. Once again we can specify a residual equation for the error in the smoothness relation.

$$\xi_{i,j}{}^S = [f(p_{i,j}, q_{i,j}, p_{i-1,j}, q_{i-1,j}, p_{i+1,j}, q_{i+1,j}, p_{i,j-1}, q_{i,j-1}, p_{i,j+1}, q_{i,j+1}, \ldots)]^2 \quad ,$$

where f is the relationship between the surface orientation at (i,j) and its neighbors. An example of the type of relationship is the difference between the surface orientation of pixel (i,j) and the mean value of the surface orientations of its 4-neighbors.

We have two constraints that need to be satisfied simultaneously, one from image irradiance and one from surface smoothness. At each stage of the iterative process, the total residual error for pixel (i,j) can be described by

$$\xi_{i,j} = \lambda \xi_{i,j}{}^R + \xi_{i,j}{}^S \quad ,$$

where λ is a weighting factor that can adjust[1] the influence of the error in image irradiance to the error in smoothness. For the image, the total residual error is

$$\xi = \sum_{i,j} \xi_{i,j} \quad .$$

The allocation of surface orientations to all pixels should minimize this total error, that is,

$$\frac{\partial \xi}{\partial p_{i,j}} = 0 \quad ,$$

[1]Since the error in image irradiance is not necessarily commensurate with that in surface smoothness some form of normalization is required.

$$\frac{\partial \xi}{\partial q_{i,j}} = 0 \quad .$$

Differentiating ξ with respect to $p_{i,j}$ and also with respect to $q_{i,j}$ gives two equations for each pixel in the image. While complicated forms of the relationship between $p_{i,j}, q_{i,j}$ and their neighboring pixels will generally occur we choose our smoothness relation so that we can arrange the equations in the form

$$p_{i,j} = F_1(p_{i,j}, q_{i,j}, \mathit{and\ p's\ and\ q's\ of\ neighboring\ pixels}) \quad ,$$

$$q_{i,j} = F_2(p_{i,j}, q_{i,j}, \mathit{and\ p's\ and\ q's\ of\ neighboring\ pixels}) \quad ,$$

where F_1, and F_2 are functions.

We therefore have an iterative scheme that, given some initial solution, we improve by reducing the residual error in image irradiance and surface smoothness. We need to ask the following questions of such a scheme: Under what conditions will it converge to a solution? Is that solution unique? Can boundary conditions be used as well? Does smoothness, as defined by our relation, give us the type of surface we want?

3 SURFACE ORIENTATION

There are many equivalent parameterizations of surface orientation. Mentioned previously were the parameters p and q, the derivatives of depth with respect to image coordinates. Some authors prefer to specify surface orientation, using slant and tilt. Slant is the angle between the surface normal and the viewing direction, while tilt is the angle between the image x axis and the projection of the surface normal onto the image plane. Other parameterizations [7] have been used when particular properties of the parameterization are to be exploited. The parameters we use are l and m:

$$l = \sin\sigma \cos\tau \quad ,$$
$$m = \sin\sigma \sin\tau \quad ,$$

where σ is the surface slant and τ its tilt, l is the component of the surface normal in the direction of the x axis and m is the component in the y direction. We select this particular parameterization, as l and m are bounded:

$$0 \le l^2 + m^2 \le 1 \quad .$$

For surfaces that we can see,

$$0 \le \sigma < \frac{\pi}{2} \quad ,$$

$$0 \le \tau < 2\pi \quad .$$

Consequently l and m specify the surface normal of an imaged surface without ambiguity.

Sometimes it is useful to think of the orientation parameterization as being obtained by mapping points on the Gaussian sphere onto a transformation plane. l and m space is the disc obtained by orthographically projecting onto an equatorial plane the points on the hemisphere of the Gaussian sphere representing those surfaces that can be seen.

4 FORMULATIONS USED FOR SURFACE RECOVERY

To explore the issues of convergence, propagation of boundary conditions, and the type of surface smoothness promotes, we formulate the problem in two ways: parallelling the technique previously described and, alternatively, resembling the use of the relaxation method to solve structural engineering problems.

The function for scene radiance, used to create synthetic images for the experiment and used by the shape recovery algorithms, is

$$R(l,m) = 0.1569\left(\frac{1+\sqrt{1-l^2-m^2}}{2}\right) + Max[0.4437\sqrt{1-l^2-m^2}+0.3137l+0.3137m, 0].$$

This function is appropriate for a scene exhibiting Lambertian reflectance and which is illuminated by both a collimated source and a uniform hemispherical source (atmosphere). The particular numerical constants specify the light direction and intensity, and the surface albedo.

The first formulation is similar to that described previously; we shall call this the 'usual' formulation. From the image irradiance equation we have the error term

$$\xi_{i,j}{}^R = (I_{i,j} - R_{i,j})^2 \quad .$$

The smoothness constraint is the requirement that $l_{i,j}$ be the average of its 4-neighbors, and $m_{i,j}$ be the average of its 4-neighbors. The error term for smoothness is

$$\begin{aligned}\xi_{i,j}{}^S = \; & (l_{i,j} - \frac{l_{i-1,j}+l_{i+1,j}+l_{i,j-1}+l_{i,j+1}}{4})^2 \\ & +(m_{i,j} - \frac{m_{i-1,j}+m_{i+1,j}+m_{i,j-1}+m_{i,j+1}}{4})^2 \quad .\end{aligned}$$

Note that this constraint is exact for a surface that is locally spherical, i.e., has equal curvature along orthogonal directions.

Minimizing $\xi = \sum_{i,j} \lambda \xi_{i,j}{}^R + \xi_{i,j}{}^S$ by differentiating with respect to $l_{i,j}$, and with respect to $m_{i,j}$, and then setting each result equal to zero, we obtain the expressions

$$\begin{aligned} l_{i,j} = \; & 0.4(l_{i-1,j}+l_{i+1,j}+l_{i,j-1}+l_{i,j+1}) - \\ & 0.1(l_{i-1,j-1}+l_{i+1,j+1}+l_{i-1,j+1}+l_{i+1,j-1}) - \\ & 0.05(l_{i-2,j}+l_{i+2,j}+l_{i,j-2}+l_{i,j+2}) + \\ & 0.8\lambda(I_{i,j}-R_{i,j})\frac{\partial R}{\partial l}\,|_{i,j} \quad , \\ m_{i,j} = \; & 0.4(m_{i-1,j}+m_{i+1,j}+m_{i,j-1}+m_{i,j+1}) - \\ & 0.1(m_{i-1,j-1}+m_{i+1,j+1}+m_{i-1,j+1}+m_{i+1,j-1}) - \\ & 0.05(m_{i-2,j}+m_{i+2,j}+m_{i,j-2}+m_{i,j+2}) + \\ & 0.8\lambda(I_{i,j}-R_{i,j})\frac{\partial R}{\partial m}\,|_{i,j}\end{aligned}$$

We use these as our iterative scheme to improve on an intial solution.

The other formulation we use, the 'engineering' formulation, creates error terms from the image irradiance equation and the smoothness constraints, but does not combine these into one term.

$$\xi_{i,j}{}^R = (I_{i,j} - R_{i,j}) \quad ,$$

$$\xi_{i,j}{}^{S_1} = (l_{i,j} - \frac{l_{i-1,j}+l_{i+1,j}+l_{i,j-1}+l_{i,j+1}}{4}) \quad ,$$

$$\xi_{i,j}{}^{S_2} = (m_{i,j} - \frac{m_{i-1,j}+m_{i+1,j}+m_{i,j-1}+m_{i,j+1}}{4}) \quad .$$

We view the ξ's as residuals and apply the relaxation approach of reducing the largest residuals. If $\xi_{i,j}{}^{S_1}$ or $\xi_{i,j}{}^{S_2}$ is selected for reduction we choose to reduce both as each is independent of the other. When $\xi_{i,j}{}^R$ is chosen for reduction, we do the reduction in two stages, - one stage altering $l_{i,j}$ and the other $m_{i,j}$. Of course, we can scale the residuals, reduce them from, say, the image irradiance equation to a certain level before introducing smoothness, vary the amount of correction we apply, (e.g. we can overrelax) and the like. In fact we can experiment with various relaxation approaches. In this formulation major changes in the relaxation scheme generally require minor programming changes.

5 EXPERIMENTAL RESULTS

The test image, shown in Figure 1, was that of a hemisphere placed on a plane, i.e., a synthetic image generated by the reflectance function previously described. The collimated light source is at slant $\frac{\pi}{4}$ and tilt $\frac{\pi}{4}$, that is, the light source is at the upper right as we view the image. We purposely avoided the case in which the collimated source is at the same position as the viewer, since the resulting symmetric reflectance map might bias the algorithm to return a symmetric surface. A synthetic image of a sphere was selected as the test image because both the image irradiance equation and the smoothness relationship we use hold exactly.[2] The performace of the algorithm to recover the surface shape could be assessed without the complications involved in using inexact models for reflectance and smoothness.

We need initial solutions to start our iterative/relaxation procedures. We used four sets of initial conditions: (1) a plane perpendicular to the viewing direction; (2) a plane slanted $\frac{\pi}{4}$ to the viewing direction; (3) a cone with its axis in the viewing direction; (4) the correct solution perturbed by small random errors.

Previous work has used boundary conditions to constrain the recovered surface. Investigating this approach we constrained the surface in various ways: at the edge of the hemisphere, at a closed curve lying on the sphere's surface, or at individual points on the sphere's surface. We also used the algorithms without any boundary conditions whatsoever.

Since we wished to investigate the ability of smoothness to propagate boundary conditions, we used various image quan-

[2]The smoothness relationship does not hold at the edge of the hemisphere where it joins the plane.

tizations, namely 16 $\times$ 16, 32 $\times$ 32, and 64 $\times$ 64.

The findings can be characterized as follows:

- Both techniques - the engineering and the usual method - gave essentially the same results.
- The engineering technique converged much faster than the usual technique.
- Smoothness propagates boundary conditions by no more than a few pixels
- The initial solution largely predetermines the final solution.

Figures 2-11 display examples of the results we achieved by means of the 'usual' iterative scheme. In each of these figures the top left picture shows the profile of the recovered surface (viewed from the bottom left corner), while at the top right we find an image that is the sine of the surface slant, with black representing 0 and white 1. The bottom left is the cosine and the bottom right the sine of the surface tilt, with black representing -1, gray 0, and white +1. The results are presented in this manner so that the performance of the algorithms can be assessed. The profile can on occasion appear more accurate than the individual surface orientations (as might be expected of an integration procedure); on other occasions, however, errors in the surface orientation (sometimes just from the image quantization) of highly slanted surfaces cause the integration routine that produces the surface shape for profiling to overstate the error. Figure 2 shows the results that should be obtained if the shape recovery algorithms recovered the surface exactly.

Figures 3-6 illustrate the effects of various boundary conditions. The errors at the edge of the sphere, where it joins the plane are expected, as smoothness does not hold there. Each figure is the result of 320 iterations, this being five times the linear dimension of the picture used. The boundary condition at a point affects an area of approximately 10 pixels in radius. Only for Figure 6, where a random five percent of pixels were set to their correct values, is the surface shape recovered correctly. Smoothness as a propagator affects but a small area. Figures 4,7, and 8 further illustrate this point. Here various image sizes are used. Observe that, as the image size increases, the boundary conditions have less effect and the solution becomes progressively worse. Figures 4,9, and 10 show the dominant influence of the initial solution. Figure 11 is included to show the effect of smoothness when $\lambda = 0$ - namely, when image irradiance does not affect the solution at all. This figure, obtained after 320 iterations demonstrates what smoothness alone can achieve, even when the definition of smoothness is exact for the viewed scene (a sphere).

Smoothness is a poor selector of surface shape and a poor propagator of boundary information when it is used to tie the surface orientation of a particular surface point to those of its neighbors. Generally, in engineering, problems solved with relaxation techniques are formulations that relate a given property at one point to that of its neighbors by means of differential relations. It is the derivative that propagates boundary information and selects a particular solution to be recovered. Following, we present such a formulation in an attempt to relieve smoothness of its role as propagator and selector.

6 PROPERTIES OF SURFACE RADIANCE

Our formulation of the relationship between image irradiance and scene radiance is

$$I(x,y) = R(l,m) \quad ,$$

where $I(x,y)$ is the image irradiance at image point x,y and $R(l,m)$ is the scene radiance for a surface normal we represent by l,m. Note that we have not selected a particular image projection in formulating this equation. l and m are the surface normals relative to the viewing direction for that surface patch. If we are using a projective transformation, l and m calculated for image point x,y will then have to be adjusted for the projective distortion; if we are using orthographic projection, zero adjustment will be required.

R is a function of the components of the surface normal and they in turn are functions of image coordinates. $R(l,m)$ specifies the relationship between surface radiance and surface orientation, while $l(x,y)$ and $m(x,y)$ specify the relationship between surface orientation and image coordinates. $R(l,m)$ embodies knowledge of the nature of surface reflection, while $l(x,y)$ and $m(x,y)$ embody the surface shape.

To provide the additional constraints we need for relating surface orientation to image irradiance, we introduce axioms that relate properties of $R(l,m)$, - that is, the constraints specify the relationship between surface radiance and surface orientation. Our axioms are

$$(1-l^2)R_{ll} = (1-m^2)R_{mm} \quad ,$$
$$(R_{ll} - R_{mm})lm = (l^2 - m^2)R_{lm} \quad ,$$

where R_{ll} is the second partial derivative of R with respect to l, R_{mm} is the second partial derivative of R with respect to m, and R_{lm} is the second partial cross derivative of R with respect to l and m. Pentland has pointed out[3] that if one interprets the image of a unit sphere as the reflectance map for a Lambertian surface illuminated in the same manner as the sphere, then (i) his result [8] is the second axiom, and (ii) the first axiom may be derived from his equations. We choose to take these relationships as axioms rather than use other assumptions.

These axioms do not have to be true embodiments of the physical laws of nature; rather, they represent the beliefs a visual system has regarding the physical laws of nature. In circumstances in which such beliefs do not hold the visual system will make errors in predicting the true nature of the world. Of course, if these axioms are not good approximations for the physical laws of nature, the visual system embodying them is useless.

Both axioms hold for the following particular forms for scene radiance.[4]

[3]Personal communication.

[4]Except at a self-shadow edge, where $R(l,m)$ is not differentiable.

(i) Lambertian reflectance: illumination by a uniform hemispherical source (for example 'sky' illumination)

$$R(l,m) = a\left(\frac{1+\sqrt{1-l^2-m^2}}{2}\right) \quad ,$$

where a is a constant, a function of the illumination strength and surface albedo.

(ii) Lambertian reflectance: illumination by a single collimated source (for example infinite point source)

$$R(l,m) = Max[0, a(b\sqrt{1-l^2-m^2} + cl + dm)] \quad ,$$

where a is a constant, a function of the illumination strength and surface albedo, and b,c, and d are constants, functions of the position of the illumination source.

(iii) Lambertian reflectance; illumination by multiple collimated sources plus a uniform hemispherical source

$$R(l,m) = a\left(\frac{1+\sqrt{1-l^2-m^2}}{2}\right) + \sum_i Max[0, a_i(b_i\sqrt{1-l^2-m^2} + c_il + d_im)] \quad ,$$

where i denotes the source.

If an extended light source is considered as comprising a large set of collimated sources at different positions with different directions and illumination strengths, then extended sources are included in this case.

(iv) Scene radiance functions that are linear functions of the components of the unit surface normal

$$R(l,m) = e\sqrt{1-l^2-m^2} + fl + gm \quad ,$$

where e,f, and g are constants. Note that $\sqrt{1-l^2-m^2}$ is the component of the unit surface normal in the direction perpendicular to the image plane.

For some forms of the scene radiance expression, only one axiom holds. This is the situation for images produced by a scanning electron microscope. The expression for scene radiance [7] is

$$R(l,m) = \frac{a}{2}\left(1 + \frac{1}{\sqrt{1-l^2-m^2}}\right) \quad .$$

The first axiom does not hold but the second $(R_{ll} - R_{mm})lm = (l^2-m^2)R_{lm}$ does. Note that $\frac{lm}{l^2-m^2} = \frac{\tan 2\tau}{2}$ so that the second axiom is about surface tilt. The first axiom introduces slant. In using the equations (that we are yet to derive) to recover surface orientation one might anticipate that they would predict tilt correctly for the surfaces in electron microscope images, but err in predicting slant.

For other forms of the scene radiance expressions neither axiom holds. Specular reflectance has been approximated [2] by

$$\begin{aligned} R(l,m) = \quad & a[b(1-l^2-m^2) \\ & + cl\sqrt{1-l^2-m^2} \\ & + dm\sqrt{1-l^2-m^2}]^n \quad , \end{aligned}$$

where n is a constant, usually having a value between 1 and 10 that determines the 'sharpness' of the specular peak.

For the maria of the moon the form of scene radiance usually used [2] is

$$R(l,m) = \frac{a(bl+cm)}{\sqrt{1-l^2-m^2}} \quad .$$

a,b,c, and d in the above expressions are constants associated with the strength and position of the light source, and with the surface albedo.

The axioms do not hold in either of the preceding cases. We would expect a visual system embodying them to make errors under these circumstances. Nevertheless this should not induce us to immediately begin searching for new axioms. After all the human visual system is not perfect under conditions of specular reflection; moreover, people observed the moon throughout history without concluding that it was spherical.

If these axioms are embodied in the human visual system, the predictions based on them - i.e. when the visual system will return 'correct' and 'incorrect' information - could be tested by psychological experiments.

We investigate the relationship between image irradiance and surface shape when both these axioms hold.

7 SURFACE ORIENTATION - IMAGE IRRADIANCE EQUATIONS

Differentiating

$$I(x,y) = R(l,m)$$

with respect to x and y, we obtain

$$I_x = R_l l_x + R_m m_x \quad ,$$

$$I_y = R_l l_y + R_m m_y \quad ,$$

$$I_{xx} = R_{ll}l_x{}^2 + R_{mm}m_x{}^2 + 2R_{lm}l_xm_x + R_l l_{xx} + R_m m_{xx} \quad ,$$

$$I_{yy} = R_{ll}l_y{}^2 + R_{mm}m_y{}^2 + 2R_{lm}l_ym_y + R_l l_{yy} + R_m m_{yy} \quad ,$$

$$\begin{aligned} I_{xy} = R_{ll}l_xl_y + R_{mm}m_xm_y + R_{lm}(l_xm_y + l_ym_x) \\ + R_l l_{xy} + R_m m_{xy} \quad , \end{aligned}$$

where subscripted variables denote partial differentation with respect to the subscript(s).

From the axioms we derive the relationships[5]

$$R_{ll} = \frac{1-m^2}{lm}R_{lm} \quad ,$$
$$R_{mm} = \frac{1-l^2}{lm}R_{lm} \quad .$$

[5]Or an equivalent set of axioms.

Substituting these relationships for R_{ll} and R_{mm} in the expressions for I_{xx},I_{yy},and I_{xy}, we obtain

$$[l_x{}^2(\frac{1-m^2}{lm}) + m_x{}^2(\frac{1-l^2}{lm}) + 2l_x m_x]R_{lm} = I_{xx} - R_l l_{xx} - R_m m_{xx} \quad ,$$
$$[l_y{}^2(\frac{1-m^2}{lm}) + m_y{}^2(\frac{1-l^2}{lm}) + 2l_y m_y]R_{lm} = I_{yy} - R_l l_{yy} - R_m m_{yy} \quad ,$$
$$[l_x l_y(\frac{1-m^2}{lm}) + m_x m_y(\frac{1-l^2}{lm}) + l_x m_y + l_y m_x]R_{lm} = I_{xy} - R_l l_{xy} - R_m m_{xy} \quad .$$

By removing R_{lm},and substituting the expressions for R_l and R_m defined by the expressions for I_x and I_y, we produce two partial differential equations relating surface orientation to image irradiance:

$$\alpha\theta l_{xx} + \beta\theta m_{xx} - \alpha\gamma l_{xy} - \beta\gamma m_{xy} = \chi\theta I_{xx} - \chi\gamma I_{xy} \quad ,$$
$$\alpha\theta l_{yy} + \beta\theta m_{yy} - \alpha\delta l_{xy} - \beta\delta m_{xy} = \chi\theta I_{yy} - \chi\delta I_{xy} \quad ,$$

where

$$\alpha = I_x m_y - I_y m_x \quad ,$$
$$\beta = I_y l_x - I_x l_y \quad ,$$
$$\gamma = l_x{}^2(1-m^2) + m_x{}^2(1-l^2) + 2l_x m_x lm \quad ,$$
$$\delta = l_y{}^2(1-m^2) + m_y{}^2(1-l^2) + 2l_y m_y lm \quad ,$$
$$\theta = l_x l_y(1-m^2) + m_x m_y(1-l^2) + (l_x m_y + l_y m_x)lm \quad ,$$
$$\chi = l_x m_y - l_y m_x \quad .$$

These equations are derived by using the two axioms and assumptions of differentiablity of the surface and of the image intensities. They represent the relationship between surface orientation and image irradiance as percieved by a visual system whose beliefs regarding the physical laws of nature are described by the two axioms.

8 SPECIAL CASES

The two partial differential equations, while derived without assumptions about the surface shape, are complex and difficult to use for the process of surface reconstruction from image intensities. Simplifying assumptions, which are naturally more restrictive, can be made to reduce the complexity of the equations and hence make them more tractable for the task of surface reconstruction. Three cases of approximation are presented: 'chain mail' restricted 'chain mail', and spherical. The spherical approximation is free of surface derivatives, the restricted chain mail introduces curvature in a limited manner, and the chain mail adds rate of change of curvature in a limited manner. Notice that the restrictions all refer to surface shape.

The chain mail approximation is so called because we imagine the surface as being covered with small flat plates hinged together, so that the axes of the hinges are parallel to the x and y axes of the image. The component of the surface normal in the x axis direction will not vary in the y direction; that is, the hinge is rigid and similarly, the component in the y axis direction will not vary in the x direction. We have $l_y = 0, m_x = 0$; consequently $l_{yy} = 0, m_{xx} = 0, l_{xy} = 0$, and $m_{xy} = 0$. The two partial differential equations reduce to

$$I_x m_y l_{xx} = l_x m_y I_{xx} - l_x{}^2(\frac{1-m^2}{lm})I_{xy} \quad ,$$
$$I_y l_x m_{yy} = l_x m_y I_{yy} - m_y{}^2(\frac{1-l^2}{lm})I_{xy} \quad .$$

The restricted chain mail approximation further requires that the 'curvature' of the chain mail be constant. Applying the additional restrictions $l_{xx} = 0$, and $m_{yy} = 0$, we obtain the equations

$$\frac{1-l^2}{lm} = \frac{l_x}{m_y}\frac{I_{yy}}{I_{xy}} \quad ,$$
$$\frac{1-m^2}{lm} = \frac{m_y}{l_x}\frac{I_{xx}}{I_{xy}} \quad .$$

The spherical approximation assumes that we are on a spherical surface. Besides the restrictions in the restricted chain mail approximation, a spherical surface implies $l_x = m_y$, - that is constant curvature independent of direction. For this case the partial differential equations become relationships between image irradiance and the direction of the surface normal:

$$\frac{1-m^2}{lm} = \frac{I_{xx}}{I_{xy}} \quad ,$$
$$\frac{1-l^2}{lm} = \frac{I_{yy}}{I_{xy}} \quad .$$

These results for the spherical approximation are equivalent to those Pentland was able to obtain [8] through local analysis of the surface. Using this technique, the only assumptions he needs is that the surface's principal curvatures are locally equal.

These special cases are presented because the resulting equations are less complex that the general ones. In the recovery of surface shape from image irradiance data, initial solutions (obtained under more restrictive assumptions) are often necessary.

9 RECOVERY OF SURFACE SHAPE

It is difficult to use the surface orientation - image irradiance equations to recover surface shape from image intensities. Since an analytic form for the image intensities is not available, numerical procedures must be employed. Two types of approaches are possible. The two differential equations can be integrated in a step-by-step manner or, given some initial solution (possibly the spherical approximation), a relaxation procedure may be employed. The difficulities posed by these approaches are usually instability on the one hand and lack of convergence, on the other. Currently we are applying this formulation to shape recovery.

10 CONCLUSION

Axiomitization of the beliefs a visual system has about the laws of surface radiance can provide sufficient constraints to relate the irradiance of an image to the corresponding scene's surface orientations. While these axioms specify a class of functions that include many that describe known situations, it remains to characterize the class specified by the axioms. The axioms are presented without justification except that they include common situations. The axioms need to be justified on the basis of physical principles such as rotational invariance, and the physics of surface reflection. While the derived equations specify the relationship between surface orientation and image irradiance, their use in shape recovery still needs to be demonstrated.

ACKNOWLEDGEMENT

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract Nos. MDA903-79-C-0588 and DAAG29-79-C-0216; these contracts are monitored by the U.S. Army Engineer Topographic Laboratory and by the U.S. Army Research Office.

REFERENCES

1. Horn, B.K.P., Obtaining shape from shading information, in: Winston, P.H. (Ed.), *Psychology of Computer Vision*, McGraw-Hill, New York, 1975 115-155.
2. Horn, B.K.P., Understanding image intensities, *Artificial Intelligence* **8** (1977) 201-231.
3. Woodham, R.J., A cooperative algorithm for determining surface orientation from a single view, *Proceedings of the Fifth IJCAI*, Cambridge, Massachusetts, 1977 635-641.
4. Woodham, R.J., Relating properties of surface curvature to image intensity, *Proceedings of the Sixth IJCAI*, Tokyo, Japan, 1979 971-977.
5. Brooks, M.J., Surface normals from closed paths,*Proceedings of the Sixth IJCAI*, Tokyo, Japan, 1979 98-101.
6. Strat, T.M., A numerical method for shape from shading from a single image, S.M. Thesis, Department of Electrical Engineering and Computer Science,Massachusetts Institute of Technology (1979).
7. Ikeuchi, K. and Horn, B.K.P., Numerical shape from shading and occluding boundaries, *Artificial Intelligence* **17** (1981) 141-184.
8. Pentland, A.P., The visual inference of shape: computation from local features, Ph.D. Thesis, Department of Psychology, Massachusetts Institute of Technology (1982).

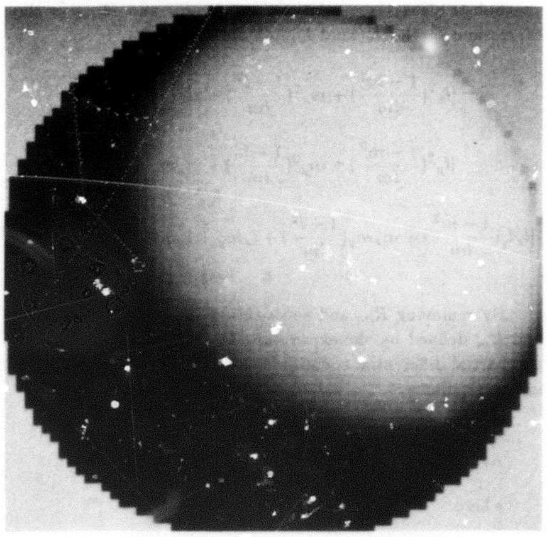


Figure 1 Original Image.

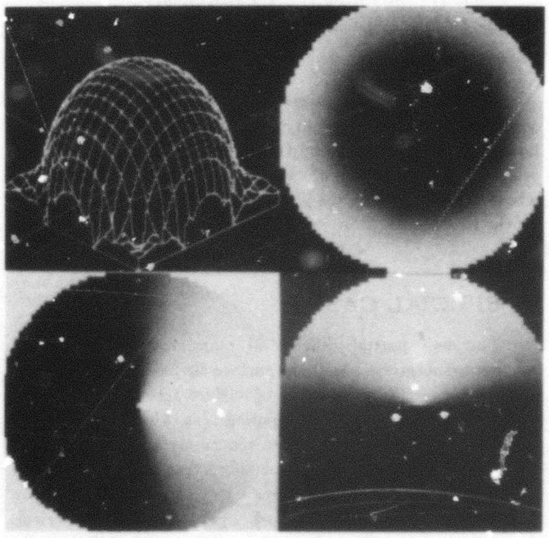


Figure 2 'Ideal' Result. Top left - profile of recovered surface; top right - sine slant, black=0, white=1; bottom left - cosine tilt, black=-1, gray=0, white=+1; bottom right - sine tilt.

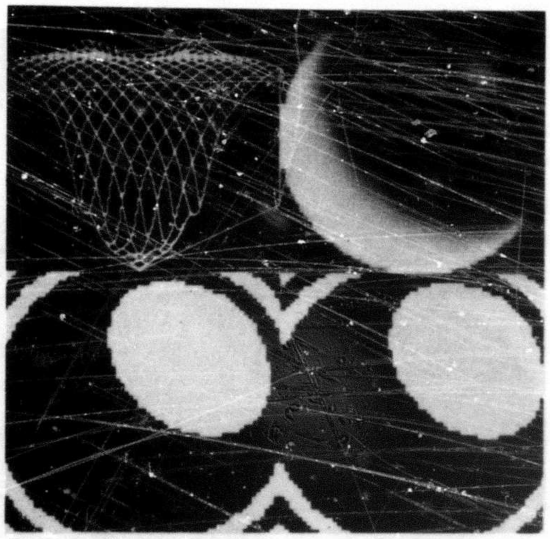


Figure 3. No boundary conditions; planar initial solution perpendicular to viewing direction; image quantization 64 X 64.

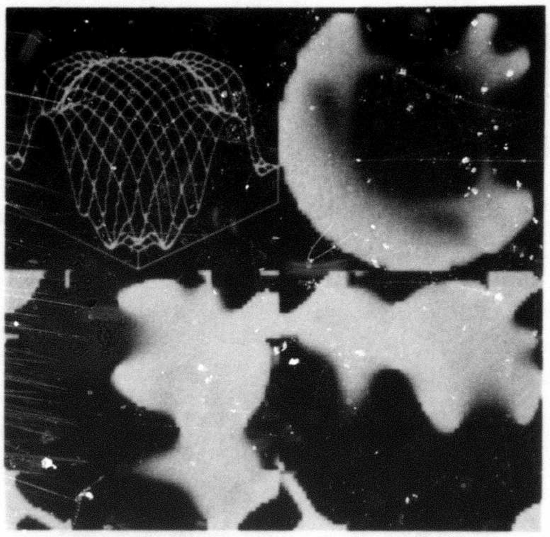


Figure 5. Boundary condition curve on sphere's surface (square shape); planar initial solution perpendicular to viewing direction; image quantization 64 X 64.

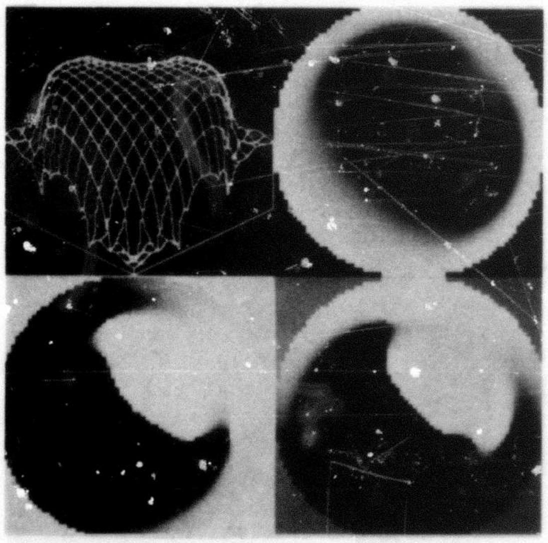


Figure 4. Boundary at edge of sphere given; planar initial solution perpendicular to viewing direction; image quantization 64 X 64.

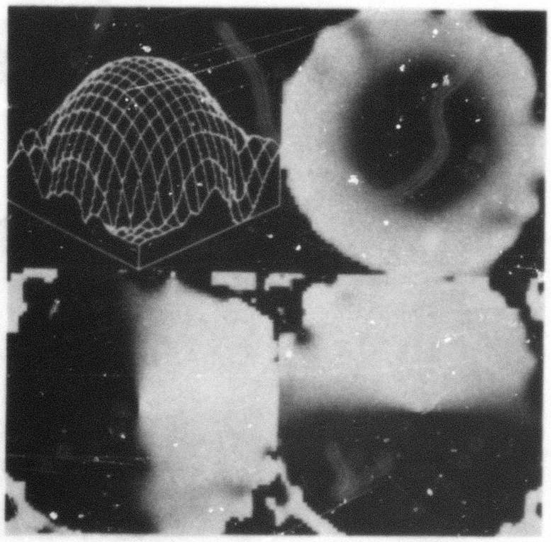


Figure 6. Random five percent of points fixed; planar initial solution perpendicular to viewing direction; image quantization 64 X 64.

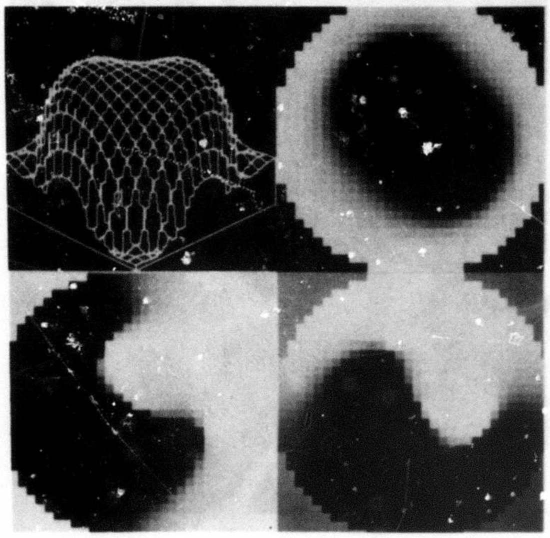


Figure 7. Boundary at edge of sphere given; planar initial solution perpendicular to viewing direction; image quantization 32 X 32.

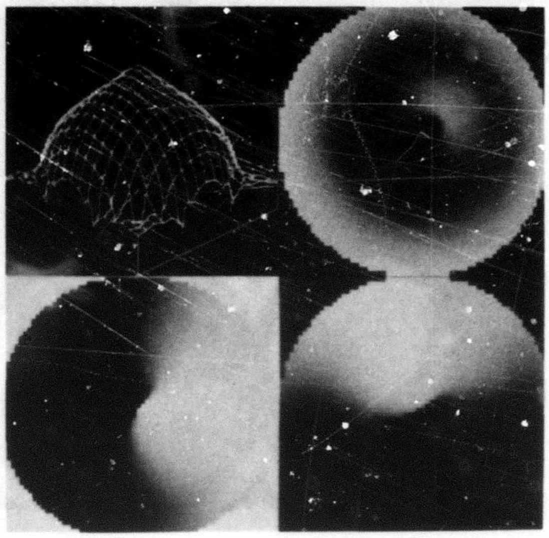


Figure 9. Boundary at edge of sphere given; cone initial solution; image quantization 64 X 64.

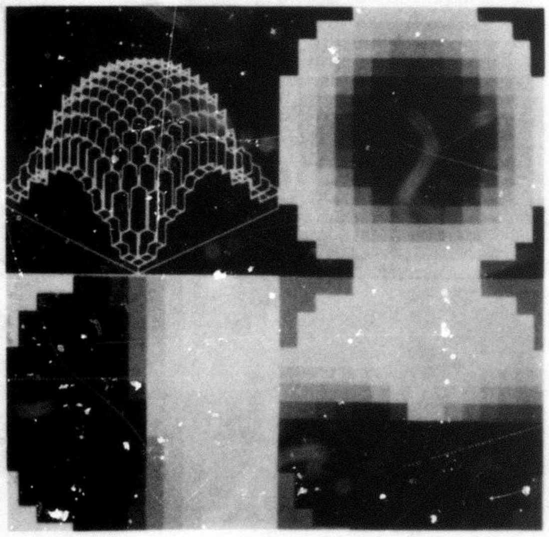


Figure 8. Boundary at edge of sphere given; planar initial solution perpendicular to viewing direction; image quantization 16 X 16.

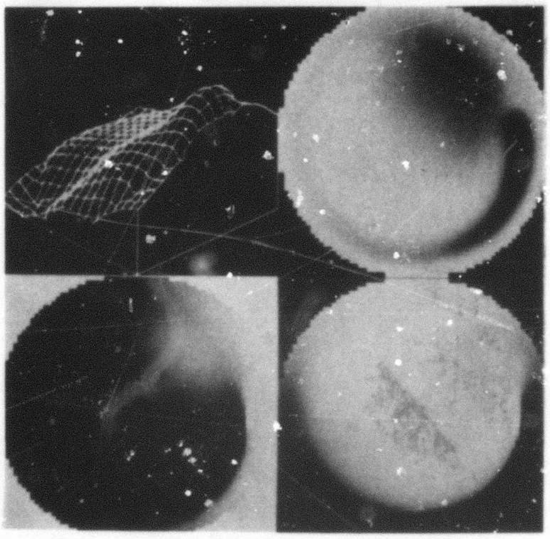


Figure 10. Boundary at edge of sphere given; planar initial solution slanted $\frac{\pi}{4}$ to viewing direction; image quantization 64 X 64.

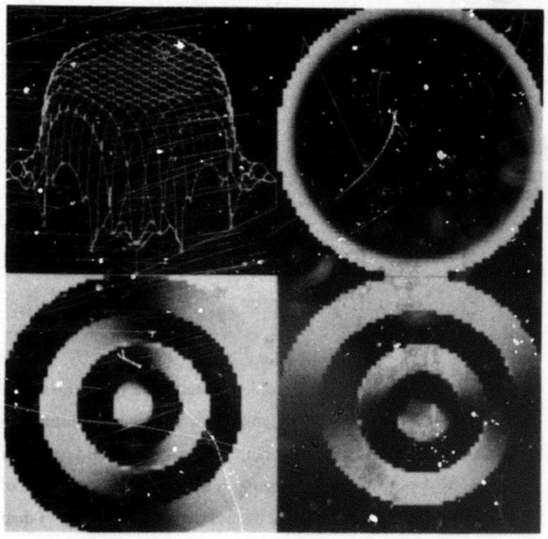


Figure 11. Smoothness constraint only. Boundary at edge of sphere given; planar initial solution perpendicular to viewing direction; image quantization 64 × 64.

AD-P000 121

CONCEPT MAPS

David M. McKeown, Jr.
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213[1]

Abstract

This paper describes a representational mechanism for constructing 3-dimensional large scale spatial organizations suitable for applications in areas such as cartography and land use studies, photo interpretation for reconnaisance and surveillance, and geological modeling for resource analysis. It focuses on the representation and utilization of map information as a knowledge source for photo-interpretation, in particular, the description of a highly detailed, large scale geographic area: Washington, D.C.. Methods of data acquisition, query specification and geometric operations on map data are discussed. These ideas have been implemented into a working map database system, CONCEPTMAP, as a component of MAPS: (Map Assisted Photo-interpretation System), our ongoing research in interactive photo-interpretation work stations.

1. Introduction

Consider the problem of building a system capable of generating answers to representative map database queries such as:[2]

- "How many bridges cross the Potomac River between Virginia and the District of Columbia."
- "Display images of National Airport before 1976."
- "What is the closest building to this geographic point."
- "Where is this geographic point."

Possible solutions to the query problem range from pre-computation and storage of potentially huge numbers of spatial relationships, to dynamic computation involving both costly search and complex geometric analysis. We favor dynamic (on demand) computation of geometric relationships, *constrained* by user defined structuring of map features and utilizing natural spatial decomposition. However, whatever the query resolution mechanism, a representation for the rich variety of man-made and natural features must underlie any such system. Further, in order to be relevent to the needs of the photo-interpreter, the results of the query should be portrayed in terms of the display of digital imagery, at the appropriate image resolution. While many queries can be answered as purely "factual" responses, (ie. *8 bridges cross the Potomac between Virgina and District of Columbia*), in our system we are able to quickly show the user the location, relative position, and scene context directly through our aerial imagery, as well as providing the necessary textual and descriptive information. This is a key point of departure from many "geographical" or "spatial information systems" [3, 4, 5]in that they simply provide tabular lookup for geographic "facts", and vector-based display of digitized map data.

Our map database consists of a collection of *concepts* each describing large spatial features such as political areas (*states, counties, towns...*), business and residential areas, parks and natural features (*rivers, streams, lakes...*). The same *concept* representation is used to hierarchically describe man-made features *airports, power stations, universities, industrial sites....* A window-oriented raster image display facility, BROWSE [9], is the man-machine interface for the concept map database and is used to create, edit, and display concept features superimposed on 2D aerial photography, and to generate arbitrary 3D scene views.

[1]This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No 3597, and monitored by the Air Force Avionics Laboratory under Contract F33615-78-C-1551. The views and conclusions in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research "rojects Agency or the U.S. Government.

[2]We are not concerned with the natural language issues of such queries, our query interface is based on a combination of query template matching, geometric specification, and interactive coordinate input through raster image display.

In this paper we will explore many of the issues raised in map feature representation and query resolution, and describe the current implementation of the CONCEPTMAP database of the Washington D.C. area. In the following section we give an overview of the MAPS system components.

2. MAPS Components

MAPS represents ongoing research in the areas of interactive aids for photo-interpretation, image/map spatial databases, and image understanding. Various components of the system have been described in [7] [*integration of terrain, imagery, and map databases*], [8] [*system goals and design*], and [9] [*BROWSE window-oriented display system*]. For completeness we will highlight the major capabilities of the MAPS components, but the reader is referred to the above for more detail.

2.1. Image Database

We have been working with an online database of approximately 40 aerial mapping photographs providing spatial and temporal overlap, centered over the Washington D.C. area. Each photograph has been digitized with a 100 micron aperature to approximately 2200x2200x8bits/pixel. Original photography scales range from 1:12000 to 1:36000, and we have recently acquired and begun to integrate 20 new digitized images at 1:60000 scale into the database. Associated with each image are several files, among which are:

- **scene description file.** Contains scene and image formation information such as *camera type, aircraft platform data, geodetic corner points, digitization data*, and *source of data*.
- **correspondence file** Contains image/map correspondence points for known ground control points. This file is interactively generated and modified by the image-to-map correspondence component.
- **coefficients file.** Contains image-to-map camera calibration coefficients, error function description, and reference to the associated correspondence file.

2.2. DLMS Database

We have adapted and restructured a geodetic based polygon feature database (DLMS Level I) [1] and a digital terrain elevation database (USGS DTED) provided by the Defense Mapping Agency, to allow for efficient feature access based on geodetic coodinates or feature attributes. This database (DLMS3D) provides a fairly coarse description of major natural and cultural terrain features and is used as a background basis for 3D display of urban scene simulations.

2.3. BROWSE Window Oriented Display

BROWSE is a window-oriented display manager which supports raster image display, overlay of graphical data such as map descriptions and image processing segmentations, and the specification and generation of 3D shaded surface models. Digitized imagery from black and white and color aerial mapping photographs is displayed by BROWSE at multiple levels of resolution and allows for dynamic positioning, zooming, expansion or shrinking of the image window. Map data represented as vectors and polygons can be superimposed on the imagery through image-to-map registration. Access to collateral map databases and terrain models may be accomplished using the BROWSE graphical interface. Finally, the window representation gives a convenient communication mechanism for passing image fragments to image interpretation programs, which generally run as separate processes. The results of such processing can be returned to BROWSE for further processing by the user.

BROWSE is used regularly as a front-end for image processing and database programs in the MAPS system and also as a general purpose image display facility.

2.4. Landmark Database

A landmark database (LANDMARK) containing approximately 180 ground control points over the Washington D.C. area has been created. Each landmark entry consists of the geodetic coordinate <*latitude, longitude, elevation*>, a textual description of the landmark, and a representative image fragment which defines the ground position for the interactive user. Entries in the landmark database may be selected by *name, geodetic location*, or by *interactive menu selection* from a raster display of image fragments.

2.5. Image-to-Map Correspondence

An interactive image-to-map correspondence component (CORRES) uses the landmark database, the image database, and BROWSE window display primitives to allow a user to graphically select a landmark and indicate the corresponding point in the new image. After the specification of the first corresponding point, CORRES can generate an initial guess of the map coverage using *flight line, image scale* and *digitization data* from the scene description file stored in the image database. Landmark candidates are graphically superimposed on the new image, allowing novice users to select landmarks with little domain knowledge. Since each landmark has an associated image fragment we could extend these interactive techniques to a more semi-automatic system which would perform image fragment matching to calculate a set of local correspondence points within the landmark area, possibly resulting in a more robust match.

2.6. Hand Segmentation

An interactive human segmentation system (SEGMENT) was one of the earliest tools developed for the MAPS system. It allows the user to specify the position and shape of a map feature, as well as capabilities to edit and display segmentations in multiple levels of detail. Segmentation files are used as intermediate representations for the map database. Facilities to convert *image based* descriptions into *map based* descriptions or to project map based descriptions onto new imagery are provided.

2.7. Machine Segmentation

An experimental coarse-fine segmentation system (MACHINESEG) using region-growing and edge profile analysis has been successfully used to extract map features such as buildings, roads, and bridges from our aerial imagery. MACHINESEG uses a coarse hand or map segmentation to specify the area within which a detailed machine segmentation should be performed. The user can accept, reject or edit the segmentation descriptions as they are generated.

2.8. Feature Mensuration

A simple image based feature measurement system (PHOTOGRAM) is currently under development to provide accurate map feature ground measurement data for integration with the map database. This system uses the BROWSE window display system and the image database to calculate *linear distance*, *rectangular area*, *polygon area*, and *radial distance*.

3. Map Database

The *concept map* database component of MAPS is central to providing access to imagery, guiding photo interpretation, and processing queries about manmade and natural features. Through the image-to-map correspondence process, map knowledge can be applied to any image, and the spatial relationships of sets of imagery can be established. The concept map provides a framework within which individual map features can be associated with high-level semantic map descriptions. Concept maps capture the spatial arrangement in urban areas of neighborhoods, political, and geographical boundaries. For example, terms such as "Northwest Washington", "Georgetown", "Foggy Bottom", "Alexandria, Virginia" are often used to describe general areas within and around Washington D.C. They provide an important mechanism for symbolic access into an image database, e.g. "display images of Georgetown later than 1976". However, depicting precise boundaries of conceptual features from aerial imagery is a difficult problem. In many cases boundaries are ill-defined and highly dependent on the user's own spatial model, which often corresponds to a hierarchy of *levels of detail* among map features. The CONCEPTMAP database allows users the flexibility of describing this hierarchy in terms of a geodetic coordinate system, independent of any particular image, while using the imagery directly as the medium of input.

Concept map features can be directly used to partition large scale spatial areas based on natural spatial relationships such as *containment*, *subsumed by* and *intersection*. Using these relationships, which often arise in database queries, rather than artificial cellular or raster organizations traditionally used for spatial decomposition appears to better model the performance of human map interpreters. Additionally, as we will describe in Section 6, many queries into the map database can be resolved at the symbolic level through manipulation of spatial relationships without resorting to geometric computations. In the following section we will describe the representation and organization of concepts in the map database.

4. Concept Map Representation

Each entity in the concept map is represented by a <u>concept schema</u>. The schema is given a unique ID by the database and the user specifies a 'symbolic' print name for the concept. Each concept may have one or more <u>role schema</u> associated with it. The practical effect of multiple roles is to allow for differing views of the same geographic concept, ie., "northwest washington" has a roles of *residential area*, as well as *political* while sharing the same 3D map description. A principle role is assigned by the user, indicating a preferred view or a role whose 3D map description defines the concepts' spatial extent. Figure 1 gives the organization of the concept schema. The CONCEPTMAP database is composed of lists of concept schema, with access functions based on *symbolic name*, *geodetic coordinate* and *spatial relationships*.

4.1. Role Schema

The <u>role schema</u> depicted in Figure 2 contains the definition of a *role name* and further specification by *subrole name*, a description of *role class* (ie., buildings may be *government*, *residential*, *commercial*, etc.). The *role type* attribute addresses the issue of whether the role is *physically* realized in the scene (image), or is a *conceptual* feature such as cultural (neighborhood), political, or geographic boundaries. Further, *role type* allows the user to define a <u>role schema</u> as a collection of *aggregrate* physical or conceptual features. For example, the concept "district of columbia" has role type *aggregrate-conceptual*, with aggregrate roles, "northwest washington", "northeast washington", "southwest washington", and "southeast washington". This mechanism allows the database to *explicitly* represent concepts which are strictly

```
CONCEPT SCHEMA
------------------
|Concept ID       |  <assigned by database. unique ID>
|  Concept Name   |  <user defined 'symbolic name' string>
|  Role Count     |  <number of roles defined for concept>
|  Principle Role|  <default role interpretation for concept>
|  Role List      |  <list of role id's>
|    Role ID 1    |
|        :        |
|    Role ID n    |
------------------
```

Figure 1: Concept Schema

```
ROLE ‘ HEMA
------------------
|Role ID          |  <assigned by the database>
|  Role Name      |  <major role (bridge. airport. university...)>
|  Subrole Name   |  <further specification of role>
|  Role Class     |  <generic class (industrial. govenment...)>
|  Role Type      |  <physical or conceptual. single or collection>
|  Role Derivation|  <method by which role was added to database>
|  Role Mark      |  <internal use during database query>
|  Role 3D ID     |  <assigned by database for lat/lon/elevation desc
|  Role Defn ID   |  <assigned by database for role attribute specif
|  Aggregrate List|  <list of user-defined component roles>
|    Role ID 1    |
|        :        |
|    Role ID n    |
------------------
```

Figure 2: Role Schema

composed of other concept roles, and can be used in query resolution as a form of inheritence. That is to say, attributes such as *population of* "district of columbia" can be calculated by examining the attribute values of its aggregrate roles. Similar operations based on geometric calculation of spatial containment provide a more flexible mechanism for such analysis.

Other role schema attributes are *role derivation* and *role mark*. Role derivation accounts for the method by which the role and 3D ID descriptor were added to the concept map database. Role mark is used to mark nodes during query search, and during creation, deletion and modification. Each role schema contains a unique 3D ID which defines a set of <latitude/longitude/elevation> triples which position the role in map space. The 3D description allows for *point*, *line*, and *polygon* features as primitives, and the aggregration of primitives into more complex topologies, ie. regions with holes, discontinous lines, and point lists. Figure 3 gives a list of the current role schema attribute values.

4.2. Further Role Specification

Associated with each *role name* there is a detailed *role property template* which further specifies role context dependent attributes or the *subrole*. For instance, for the role name *residential area* the subroles may be *single family, mixed housing, apartment complex, rural.* The role property template contains slots for population, housing density, roof and tree cover as a percentage of area, and other attributes. In the absence of specification by the user, default attribute values are used, within the context of the subrole. Users may dynamically create new subroles, and use existing or newly specified attribute defaults. The addition of a new role name and associated role property template requires intervention by the system maintainer. Figure 4 gives a list of the current subrole attribute values for the roles **buildings**, **bridges**, and **airport**.

Figure 5 gives a partial list of the current concept symbolic names and associated role ids. As of this writing there are 110 concepts with 183 roles in the CONCEPTMAP database. We plan to incrementally increase the complexity of the database both in terms of number of map features represented and the richness of the underlying representation.

```
ROLES:                      16 role names:
   unknown              university           building              water
   bridge               political            road                  park
   reservoir            sports complex       airport               hospital complex
   residential area     geographic           industrial area       parking lot
ROLE-TYPES:                 5 role types:
   unknown              physical             aggregrate-physical   conceptual
   aggregrate-conceptual
ROLE-CLASS:                 8 role classes:
   unknown              industrial           transportation        natural feature
   residential          government           cultural feature      commercial
ROLE-DERIVATION:            5 derivation classes:
   unknown              hand-segmentation    landmark-description machine-segmentation
   terminal-interaction
ROLE-MARK:                  9 mark classes:
   none                 geo-query            template-query
   new-concept          new-role             modify-concept
   modify-role          new-3D               modify-3D
```

Figure 3: Role Schema Attribute Values

```
ROLE: building           State information for role (1):
Sub role file: /usri0/vdata/maps/building.dyn
   unknown                    performing arts complex     museum
   office building            railroad station            dormitory
   government building        administration              memorial
   concert hall               terminal building
ID file status: KEY: 'BULD'  Min: 1   Max: 38   Active: 32   Next: 39
*****************

ROLE: bridge             State information for role (2):
Sub role file: /usri0/vdata/maps/bridge.dyn
   unknown                    railroad                   pedestrian
   automobile
ID file status: KEY: 'BRDG'  Min: 1   Max: 8   Active: 8   Next: 9
*****************

ROLE: airport            State information for role (5):
Sub role file: /usri0/vdata/maps/airport.dyn
   unknown                    commercial                 military
   operations building        terminal                   runway
   hangars                    navigational beacons
ID file status: KEY: 'AIRP'  Min: 1   Max: 10   Active: 10   Next: 11
*****************
```

Figure 4: Subrole Attribute Values: building, road, airport

5. Some Examples

In this section we will briefly describe three sample concept map entries taken from the Washington D.C. concept map database. These examples illustrate the flexibility of the concept map representation and were created by interactive query to the database.

5.1. Map Feature Concept

Figure 6 shows a typical map feature entry in the CONCEPTMAP database. This entry, 'washington circle', (a traffic circle in the Foggy Bottom area) was created during an interactive terminal session. Figure 7 gives the *<latitude, longitude, elevation>* description for the 'washington circle' conceptmap entry which is defined in the role schema as 'D3ID3' and was created by interactive specification of image points in database image 'dc38617'. Using the image-to-map correspondence for 'dc38617', geodetic coordinates are calculated. Ground elevations are calculated by lookup and interpolation from our digital terrain elevation database [1]. The original image coordinates are saved for possible refinement, and are accessable through the 'D3ID' attribute.

Figure 8 gives the conceptmap entry for 2D feature description for 'washington circle'. Simple shape features such as *centroid, area, perimeter*, and *fourier coefficients* are calculated from the role schema D3ID in map coordinate space and are used by our MACHINSEG system in conjunction with the D3ID to specify location and shape of map features.

```
CONCEPT1        tidal basin  WATER1
CONCEPT2        district of columbia  POLI1  RESI2
CONCEPT3        northwest washington  POLI2  RESI1
CONCEPT4        macmillian reservoir  RESV1
CONCEPT5        southwest washington  POLI3
CONCEPT6        northeast washington  POLI4
CONCEPT7        virginia  POLI5
CONCEPT8        maryland  POLI6
CONCEPT9        kennedy center  BULD1  BULD8
CONCEPT10       ellipse  PARK1
CONCEPT11       washington circle  ROAD1
CONCEPT12       state department  BULD2
CONCEPT13       executive office building  BULD3
CONCEPT14       white house  BULD4
CONCEPT15       treasury building  BULD5
CONCEPT16       department of commerce  BULD6
CONCEPT17       arlington memorial bridge  BRDG1
CONCEPT18       rfk stadium  SPORT1
CONCEPT19       museum of history and technology  BULD7
CONCEPT20       key bridge  BRDG2
CONCEPT21       kutz bridge  BRDG3
CONCEPT22       george mason bridge  BRDG4
   :
   :
CONCEPT55       national airport  AIRP1  BULD17  AIRP3  AIRP4
CONCEPT57       u.s. capitol  PARK8  BULD18
CONCEPT58       alexandria  POLI7  RESI6
CONCEPT59       old town alexandria  RESI6
CONCEPT60       washington navy yard  INDU2
CONCEPT61       bolling air force base  AIRP5
CONCEPT62       andrews air force base  AIRP6
CONCEPT63       american pharmaceutical association  BULD19
CONCEPT64       national academy of sciences  BULD20
CONCEPT65       federal reserve board  BULD21
CONCEPT66       national science foundation  BULD22
CONCEPT67       civil service commission  BULD23
CONCEPT68       interior department  BULD24
CONCEPT69       district building  BULD25
CONCEPT70       lafayette park  PARK9
CONCEPT71       constitution hall  BULD26
CONCEPT72       national press building  BULD27
CONCEPT73       23rd street  ROAD9
CONCEPT74       constitution avenue  ROAD10
CONCEPT75       virginia avenue  ROAD11
CONCEPT76       c street  ROAD12
CONCEPT77       22nd street  ROAD13
   :
   :
```

Figure 5: Washington D.C. CONCEPTS and Role IDs [partial list]

```
Concept: 'washington circle'
Concept id: 'CONCEPT11'          1 roles (principle role: 0)
[0] washington circle
Role ID: 'ROAD1'                 Role Defn ID: ''
        Role name: 'road'        subrole: 'traffic circle'
        Role class: 'transportation'    type: 'physical'
        Role deriv: 'terminal-interaction'
        Role mark:  'none'
        3D Role ID: 'D3ID3'      3D Role pointer @0
```

Figure 6: Washington Circle

```
14 Points  Generic name:  'dc38617'  Feature type: 'areal'
maximum coordinate:  lat N38 54 10 (487)  lon W77 3 3 (829)
minimum coordinate:  lat N38 54 7 (62)  lon W77 2 59 (325)
point 0  elev: 16 meters  lat N38 54 9 (52)  lon W77 3 3 (829)
point 1  elev: 16 meters  lat N38 54 10 (29)  lon W77 3 3 (131)
point 2  elev: 17 meters  lat N38 54 10 (464)  lon W77 3 2 (265)
point 3  elev: 17 meters  lat N38 54 10 (487)  lon W77 3 1 (397)
point 4  elev: 17 meters  lat N38 54 10 (428)  lon W77 3 0 (529)
point 5  elev: 18 meters  lat N38 54 9 (752)  lon W77 2 59 (656)
point 6  elev: 18 meters  lat N38 54 9 (88)  lon W77 2 59 (325)
point 7  elev: 17 meters  lat N38 54 8 (101)  lon W77 2 59 (369)
point 8  elev: 16 meters  lat N38 54 7 (294)  lon W77 3 0 (227)
point 9  elev: 16 meters  lat N38 54 7 (62)  lon W77 3 1 (92)
point 10  elev: 16 meters  lat N38 54 7 (83)  lon W77 3 2 (286)
point 11  elev: 16 meters  lat N38 54 7 (555)  lon W77 3 3 (270)
point 12  elev: 16 meters  lat N38 54 8 (554)  lon W77 3 3 (715)
point 13  elev: 16 meters  lat N38 54 9 (52)  lon W77 3 3 (829)
```

Figure 7: Washington Circle 3D Database Entry

```
clockwise
area = 12.201516 square sec      perimeter = 12.704513 sec
fractional fill = 0.791007       compactness = 13.228246
centroid:              lat N38 54 8 (772)  lon W77 3 1 (627)
centroid of border:    lat N38 54 8 (771)  lon W77 3 1 (631)
length of major axis (fitted ellipse) = 4.323209 seconds
length of minor axis (fitted ellipse) = 3.587450 seconds
major angle (fitted ellipse) = 0.001446 radians (0.08 deg)
minor angle (fitted ellipse) = 1.572243 radians (90.08 deg)
fourier coefficients (order 1 to 9):
  1).  ax: 0.2383   bx: 1.7778    ay: 2.1423    by: -0.2884
  2).  ax: -0.0074  bx: -0.0017   ay: -0.0028   by: -0.0035
  3).  ax: 0.0343   bx: 0.0517    ay: 0.0509    by: -0.0117
  4).  ax: 0.0016   bx: -0.0045   ay: 0.0012    by: 0.0027
  5).  ax: -0.0029  bx: -0.0124   ay: 0.0136    by: 0.0034
  6).  ax: 0.0008   bx: -0.0088   ay: 0.0104    by: 0.0014
  7).  ax: 0.0091   bx: 0.0031    ay: 0.0108    by: 0.0031
  8).  ax: -0.0097  bx: -0.0030   ay: 0.0126    by: -0.0099
  9).  ax: -0.0036  bx: -0.0032   ay: 0.0088    by: 0.0011
```

Figure 8: Washington Circle 2D Shape Descriptors

The photograph in Figure 9 was created by CONCEPTMAP as a result of the query *"Display all images containing 'washington circle'"*. Using the BROWSE subroutine package as primitives, a display frame is created composed of windowed image fragments centered around the map feature. Once displayed, any of the windows can be manipulated using commands within CONCEPTMAP. Thus, the user can select one or more of the image fragments, *expand* the size of the window to obtain more image context, *move* a window for side-by-side comparison, *zoom* in for more detail, or *adjust* the center of the window.

5.2. Landmark Concept

Figure 10 lists the concept map entry for 'george mason bridge'. The role schema attribute *role derivation* specifies this concept as being a 'landmark-description'. When listing this role entry the concept schema attribute *concept name* is used to index into the landmark database (LANDMARK)[8] to produce the textual description which defines the landmark entry. This allows entries in our landmark database to be directly accessible through the concept map.

```
Concept: 'george mason bridge'
Concept id: 'CONCEPT22'          1 roles (principle role: 0)
[0] george mason bridge
Role ID: 'BRDG4'                 Role Defn ID: ''
        Role name: 'bridge'      subrole: 'automobile'
        Role class: 'transportation'    type: 'physical'
        Role deriv: 'landmark-description'      mark: 'unknown'
        3D Role ID: 'D3ID14'     3D Role pointer @0
latitude 38 52 43 300
longitude 77 2 22 500
elevation 12 meters
1140,1482 in /visf/washdc/asc/dc1419/1bw.img
landmark image at resolution 1

                    george mason bridge
Definition:  A bridge spanning the Potomac River in southwest DC.
Located adjacent to the Jefferson Memorial and the Rochambeau Bridge.

Description:  The George Mason Bridge, also known as one of the
twin 14th St. Bridges, carries the westbound lanes of U.S. 1
across the Potomac from 14th St. on the east bank to the Jefferson
Davis Highway on the west.  The landmark image is oriented with
north at the top.
```

Figure 10: Role Schema: George Mason Bridge

5.3. Multiple Role Concept

The concept 'national airport' is an example of a more complex organization of role schema. Figure 11 shows the current concept description for 'national airport'. The principle role AIRP1 defines this concept to be a commercial airport, whose boundary should be interpreted as a aggregrate-physical feature, that is a collection of physically realizable boundary descriptions. Within the context of the area represented in 'D3ID59' will be found all roles which comprise this concept.

The other roles define the airport terminal building, a runway, and a collection of hangars. The terminal building 'BULD17' and the airplane hangar 'AIRP4' have boundary descriptions associated with them, while the runway 'AIRP3' role has none. Geometric queries on the concept map database would find the terminal building and hangar as *contained* within the principle role of the concept 'CONCEPT55', "national airport". However, a symbolic query asking for all the roles

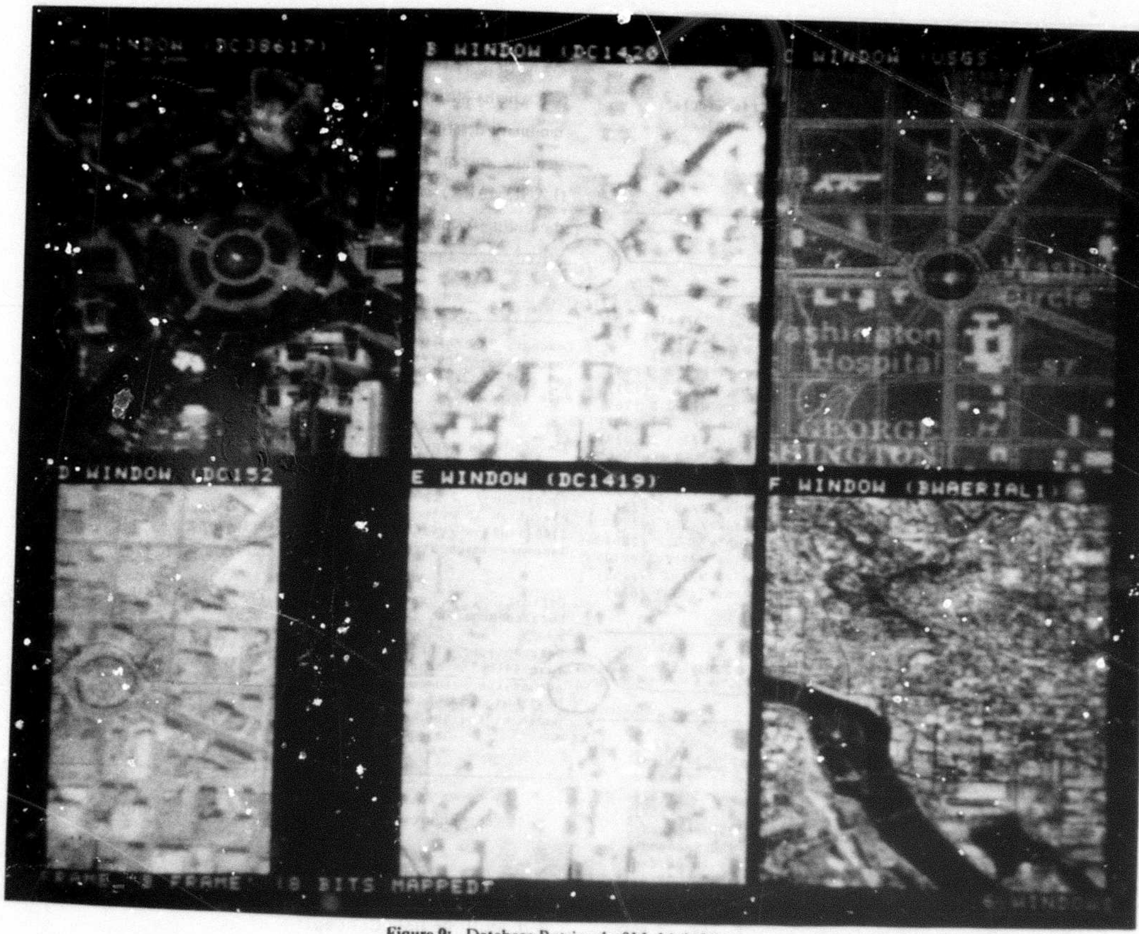


Figure 9: Database Retrieval of Multiple Views

associated with the principal role for concept 'CONCEPT55 would find the runway as well at the other two roles. The relative merits and limitations of strict user defined symbolic representation vs. using geometric queries to build plausible symbolic relationships is an area for future research.

We have begun to create detailed models for *airports* and *industrial area* roles, initially as a guide to the interactive user, but expect to integrate such static descriptions into an active query component in the future. We are calling such descriptions *site description models.* Currently, users are free to describe as role schema those portions of the airport description model that are of importance, without a requirement to create a completely specified airport *site description model.* Figure 12 details a preliminary organization of an airport description model.

We see this work as a natural development towards including in the concept map representation explicite modeling of physical relationships between site structures, functional descriptions of structures, and functional relationships (processes) between collections of model structures.

6. Map Query

There are four database access primitives which can be employed singularly or in combination to extract the positional, factual, or relational attributes required to answer the queries posed in the beginning of this paper.

```
Concept: 'national airport'
Concept id: 'CONCEPT55'          4 roles (principle role: 0)
[0] national airport
Role ID: 'AIRP1'                 Role Defn ID: ''
        Role name: 'airport'     subrole: 'commercial'
        Role class: 'transportation'    type: 'aggregrate-physical'
        Role deriv: 'hand-segmentation' mark: 'unknown'
        3D Role ID: 'D3ID59'     3D Role pointer @0
[1] national airport
Role ID: 'BULD17'                Role Defn ID: ''
        Role name: 'airport'     subrole: 'terminal'
        Role class: 'transportation'    type: 'physical'
        Role deriv: 'terminal-interaction'      mark: 'unknown'
        3D Role ID: 'D3ID60'     3D Role pointer @0
[2] national airport
Role ID: 'AIRP3'                 Role Defn ID: ''
        Role name: 'airport'     subrole: 'runway'
        Role class: 'transportation'    type: 'physical'
        Role deriv: 'hand-segmentation' mark: 'unknown'
        3D Role ID: ''  3D Role pointer @0
[3] national airport
Role ID: 'AIRP4'                 Role Defn ID: ''
        Role name: 'airport'     subrole: 'hangars'
        Role class: 'transportation'    type: 'physical'
        Role deriv: 'terminal-interaction'      mark: 'unknown'
        3D Role ID: 'D3ID61'     3D Role pointer @0
```

Figure 11: Concept Schema for National Airport

```
                               AIRPORT
        /               /         |          \              \
     /                /           |           \                \
misc.           runway tarmac  buildings   parking lots    temporal structures
  structures       |             /  |   \                    /      |   \
   / \             |     terminal   |  operations       airplanes   |  fuel trucks
fuel  navigational |           hangars                    /|\          \
tanks   beacons    |                           commercial | private    baggage trucks
                  / \                                  military
           runways   taxiways
```

Figure 12: Role Definition Template for AIRPORT Role

6.1. Signal Access

Display or list all concepts within an interactively specified image area. Signal access requires an explicite *image-to-map* correspondence. Image coordinates are used to calculate map coordinates which are used to search the concept map database. Figure 13 is a display frame created by CONCEPTMAP as a result of an interactive user query to display the area around a set of storage tanks near the Washington D.C. navy yard. The query area is superimposed as a blue overlay in each of the display windows, the area of interest is centered in each window, and displayed at the highest resolution that fits within the window partition. Signal access queries are purely dynamic, involving only the BROWSE window manager and the image database and do not use the concept map symbolic data structures.

6.2. Symbolic Access

Display or list all concepts with a given symbolic name. Requires explicite mapping of a user defined name, ie(*memorial bridge*) into the map coordinate system. As we described in section 5.1, the role schema 3D ID gives us a direct mechanism for searching the image database.

6.3. Role Template Access

Given a completely or partially specified role schema, find all roles in the concept map database which satisfy the specification. The user can specify additional constraints based on the role property template if the *role name* and *subrole name* have been specified. The result of a template access is a list of role schema ID's. These may be printed or displayed by the user as described above.

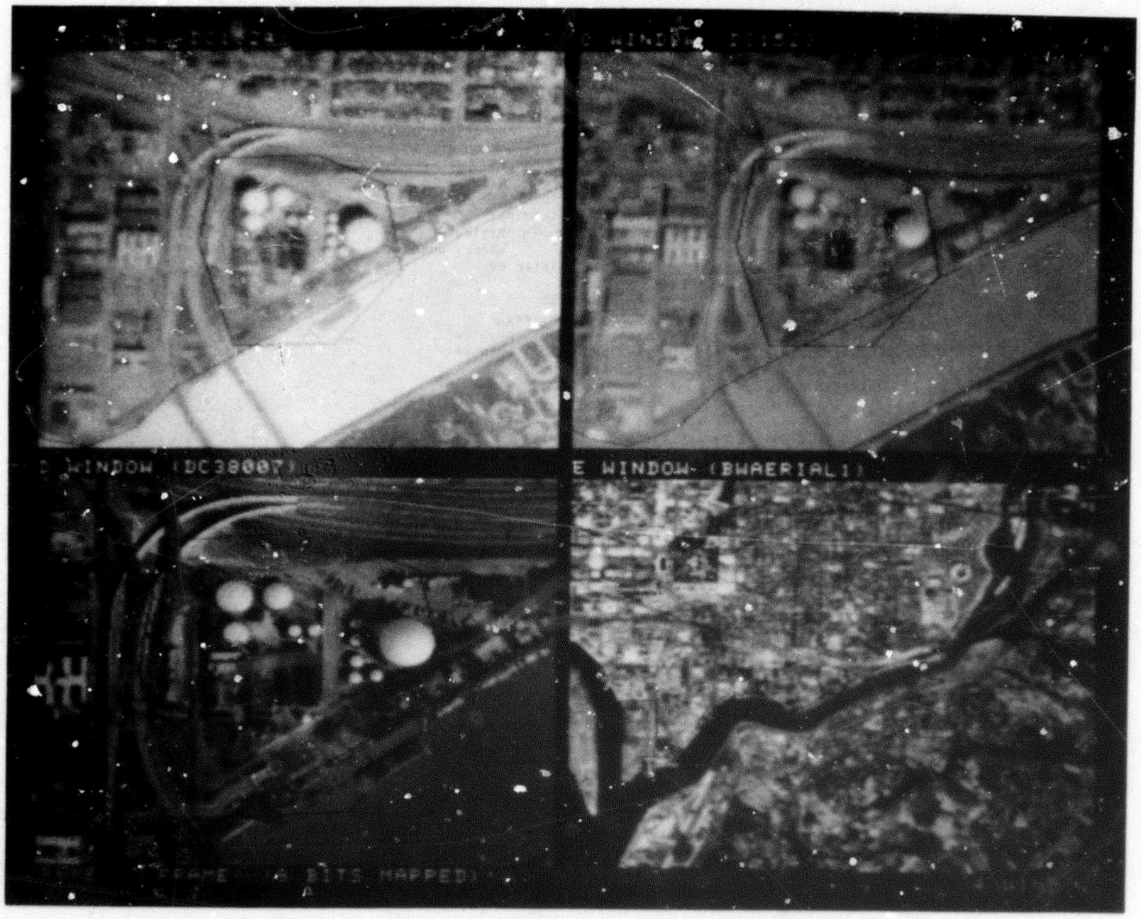


Figure 13: Signal Access Display

6.4. Geometric Access

Using the 3D role latitude/longitude/elevation description, compute geometric properties such as *containment*, *subsumed by intersection*, *adjacency*, and *closest point* A list of role schema ID's which satisfy the geometric constraints is created. In the case of *intersection* and *adjacency* a temporary role schema with 3DID is generated with the results (point, line, polygon) of the geometric operation for each pair of database role schema.

6.5. Integrating Access Methods

In order to generate answers for several of the map database queries posed at the beginning of this paper, we must actually perform sequences of symbolic, signal, template, and geometric access functions. There are clearly difference costs associated with each method, geometric computation being the most expensive, symbol to signal being the least expensive. We currently require that the user specify the sequencing of access methods, with CONCEPTMAP providing automatic storage of temporary results in the form of querylists of role schema which satisfy a primitive query. Let us analyze those sample queries in terms of our query primitives.

- "How many bridges cross the Potomac River between Virginia and the District of Columbia."
  Get symbolic level from symbolic level with template and geometric constraint

- "Display images of National Airport before 1976."
  Get image from symbolic level

- "What is the closest building to this geographic point" [*point to screen*].
  Get symbolic level from template, signal, and geometric constraint

- "Where is this geographic point." [*specify geodetic coordinate*]
  Get signal and symbolic level from signal constraint

6.6. Hierarchical Query Resolution

Consider the query "*where is the intersection of 31th and m street*". One approach is to simply find the 3D map descriptions for each of the roles (*symbol to signal*), and perform a geometric operation to calculate a ground coordinate position. The concept map provides the capability to use symbolic geometric relationships to generate the following response:

```
<greater washington d.c.>
<district of columbia>
<northwest washington>
<georgetown area>
<georgetown business district>
<lat N38 54 18 (374)   lon W77 3 41 (213)   el
```

When the actual geodetic location of an intersection is required, a geometric operation must be performed, unless it is defined as a map concept. However each of the other responses resulted from a traversal of a *containment tree* which is maintained by the concept map database. The containment tree is precomputed from all map concept features having polygonal 3DID descriptions using the geometric relationships of *subsumed by* and *contains*. Features having a clear hierarchy of *level of detail* such as political and cultural (neighborhood) concepts can form the basis for partitioning of other map features to improve query performance and to better model the spatial organization as more than just a collection of independent concepts.

For this reason, we would like to explore building hierarchical descriptions using the concept map database. We can anticipate its use as a knowledge source for more complex matching, for instance in symbolic scene recognition. For example, the occurrance of role descriptions for *oil tank farm*, *power transformers*, and *cooling towers* within close physical proximity, indicates the area may be *power plant* or *industrial*.

7. 3D Map Display

A central problem for a variety of cartographic tasks is flexible access to 3D map databases [6]. Tasks include inspection and verification of spatial databases, incremental update, and feature enhancement. CONCEPTMAP provides tools for the selection of ground area either through image-to-map correspondence (ie. describing the area to be portrayed via digital imagery) or direct specification of map coordinates. The photograph in Figure 14 shows a full frame window containing a two dimensional map image of an area around Washington D.C.. This 13 color-class thematic image[3] shows areas such as forest and park (green), water (blue), residential (yellow), and high density urban (brown). It was generated by scan conversion of a polygon map database provided by the Defense Mapping Agency (DLMS Level 1) [1].

In this application, the user indicates a rectangular area of interest in the map image, specifies the center point (west of National airport), viewing position (from the southeast), and view angle. This is done by tracking a cursor on the display to minimize the amount of knowledge that the user must have of the actual 3D coordinate system.

The photograph in Figure 15 shows the result of the 3D map generation. For each image point in the area specified by the user, a map coordinate is calculated (latitude, longitude, elevation). A 3D surface description is generated using the thematic color from the map image, and this description is passed to a 3D shaded raster graphics display program [2]. The resulting map image is then displayed by BROWSE.

The CONCEPTMAP database provides 3D map feature descriptions for the generation of cartographically accurate urban scenes. The DLMS scene as generated in 15 is used as a base map, onto which we project our map database features. The photograph in Figure 16 shows a view of the Foggy Bottom area with the observer looking towards the southeast from above the intersection of Virgina Avenue and 23rd Street. Buildings in the scene are (from left to right) *constitution hall* (clipped to the scene viewport), *interior department*, *civil service commission*, *bureau of indian affairs*, *federal reserve board*, *state department* and *national academy of science*. Roads are *virginia avenue* (bottom right to center left), *C street* (center left to middle right), and *constitution avenue* (running along the light/dark terrain boundary). The linear feature running between the *interior department* and *civil service commission* and occluded by the line of buildings in the rear of the scene is the boundary of the map description for *'foggy bottom'*.

8. Conclusions

We have discussed the current implementation of a large scale spatial map database organized around a *concept map* representation which provides for the hierarchical description of complex natural and man-made features. User defined views are supported by allowing concepts to take on multiple roles, while maintaining a consistent 3 dimensional map coordinate representation. We have shown how the CONCEPTMAP database can be used for flexible access into an image database, display of 3D urban scenes, and for query into spatial databases. We believe that this work has applications in a variety of task domains where knowledge representation can be viewed in terms of 3 dimensional spatial organizations, particularly in cartography, photo-interpretation, and geological modeling.

[3]Reproduced in stunning black and white.

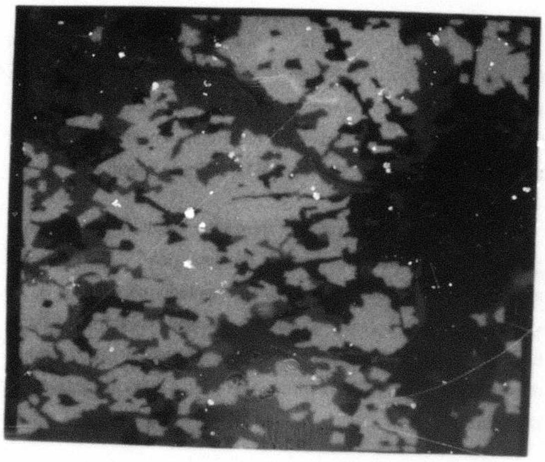


Figure 14: 2D Washington D.C. Terrain Map

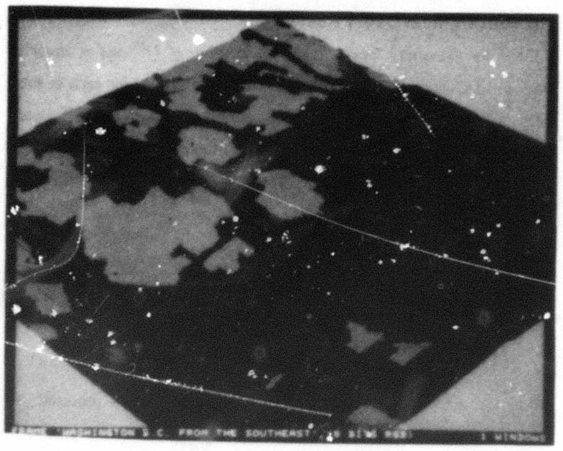


Figure 15: 3D Washington D.C. Terrain Map

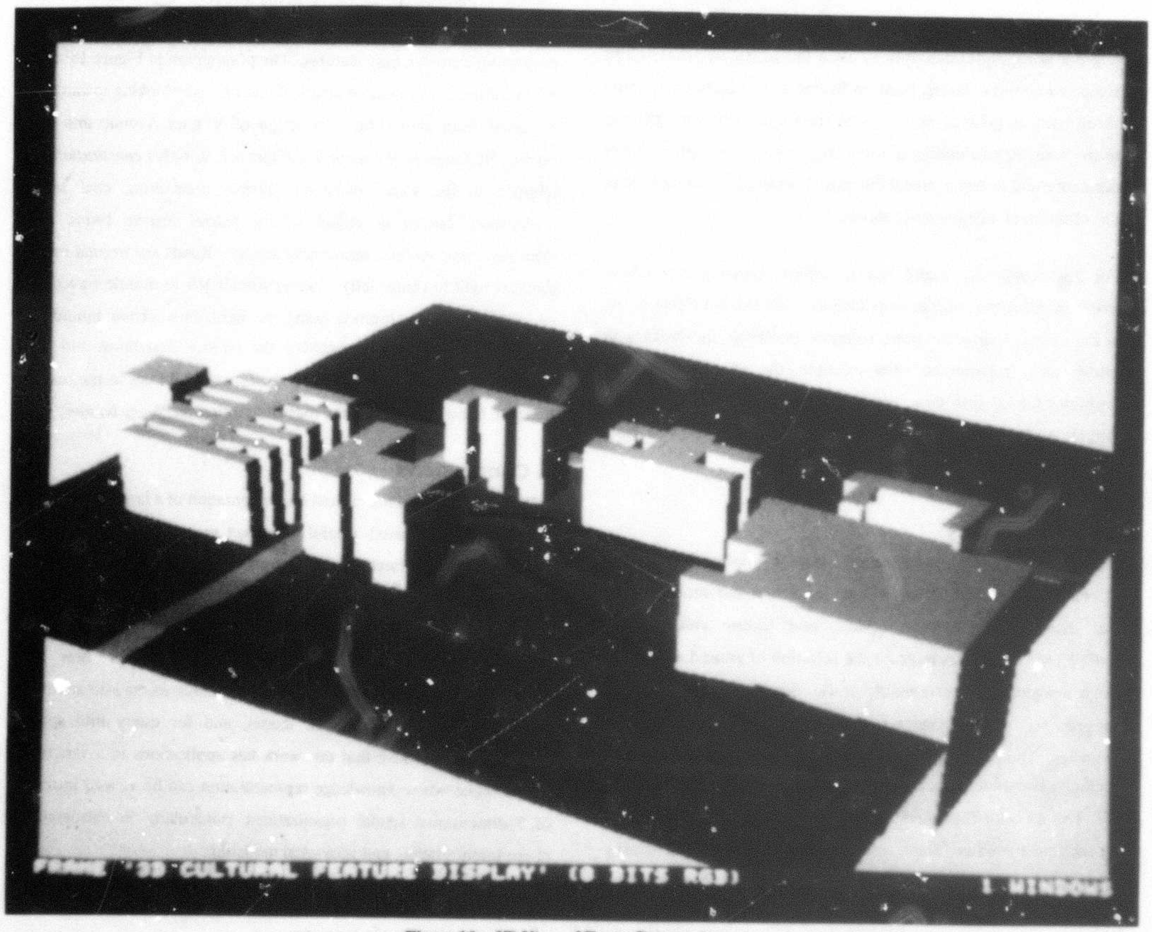


Figure 16: 3D View of Foggy Bottom Area

Acknowledgments

The MAPS implementation team is comprised of Jerry Denlinger (BROWSE system implementation), Mike Matsko (3D graphics display), and Wilson Harvey (correspondence database). Alumni members are Duane Anderson (landmark database) and Eric Grant (graphics package). Vision group members David Smith and Steve Shafer contributed to the excellent image access and user interface packages. Takeo Kanade and Raj Reddy have provided encouragement, support, and guidance for this work.

References

1. *Product Specifications for Digital Landmass System (DLMS) Database.* Defense Mapping Agency. St. Louis, Missouri, 1977.

2. Grant, Eric. AGP: A Graphics Package. Unpublished project report, Carnegie-Mellon University, Pittsburgh, PA., 1981.

3. *IEEE Workshop on Picture Data Description and Management,* Asilomar, Ca., August, 1980.

4. *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management,* Hot Springs, Va., November, 1981.

5. *IEEE Computer Magazine, Special Issue on Pictorial Information Systems,* November, 1981.

6. Lukes, G. E. Computer-assisted photo interpretation research at United States Army Engineer Topographic Laboratories (USAETL). Techniques and Applications of Image Understanding III, Society of Photo-Optical Instrumentation Engineers, Washington, D.C., April, 1981, pp. 85-94.

7. McKeown, D.M. Knowledge Structuring in Task Oriented Image Databases. Proceedings of the IEEE Workshop on Picture Data Description and Management, Asilomar, Ca., August, 1980, pp. 145-151.

8. McKeown, D. M. and T. Kanade. Database Support for Automated Photo Interpretation. Techniques and Applications of Image Understanding III, Society of Photo-Optical Instrumentation Engineers, Washington, D.C., April, 1981, pp. 192-198.

9. McKeown, D. M., and J. L. Denlinger. "Graphical Tools for Interactive Image Interpretation." *Computer Graphics 16,* 3 (July 1982), 189-198.

AD-P000 122

Rotationally symmetric operators for surface interpolation

Michael Brady and Berthold K.P. Horn

Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract

The use of rotationally symmetric operators in vision is reviewed and conditions for rotational symmetry are derived for linear and quadratic forms in the first and second partial directional derivatives of a function $f(x,y)$. Surface interpolation is considered to be the process of computing the most conservative solution consistent with boundary conditions. The "most conservative" solution is modelled using the calculus of variations to find the minimum function that satisfies a given performance index. To guarantee the existence of a minimum function, Grimson has recently suggested that the performance index should be a seminorm. It is shown that all quadratic forms in the second partial derivatives of the surface satisfy this criterion. The seminorms that are, in addition, rotationally symmetric form a vector space whose basis is the square Laplacian and the quadratic variation. Whereas both seminorms give rise to the same Euler condition in the interior, the quadratic variation offers the tighter constraint at the boundary and is to be preferred for surface interpolation.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

1. Introduction

Two separate themes from the Computer Vision literature come together in this paper: the use of rotationally symmetric operators, and the idea that several modules of visual perception require that the "most conservative" solution that meets a given set of boundary conditions be computed. The two themes are combined in an investigation of which operator to use in the interpolation of smooth surfaces from one-dimensional boundary constraints. Such constraints arise naturally in a variety of visual problems.

In the next section we review the role of rotationally symmetric operators in Computer Vision, and we derive conditions which linear and quadratic forms in the first and second directional derivatives must satisfy in order to be rotationally symmetric. We then discuss the idea that vision is a conservative process, citing examples from both figure perception and scene analysis. The "most conservative" solution is modelled using the calculus of variations to find the minimum function that satisfies a given performance index. A major problem associated with the use of the calculus of variations is guaranteeing the existence of a minimum function (see for example Courant and Hilbert 1953, p.173). A theorem of Grimson(1981, theorem 2) proves that a sufficient condition for the existence of a minimum is that the performance index should be a seminorm on the space of functions. The condition is not necessary. For example, Horn(1981) has determined the curve that minimizes the integral square curvature subject to tangency conditions at the end points; the performance index in this case is not a seminorm.

Grimson(1981) notes that many intuitively plausible performance indices based on mean and Gaussian curvature are not seminorms, but that the square Laplacian $f_{xx}^2 + 2f_{xx}f_{yy} + f_{yy}^2$ and the quadratic variation $f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2$ are. We show here that *any quadratic form* in f_{xx}, f_{xy}, and f_{yy} is a seminorm.

To further constrain the choice of performance index in the infinite set of quadratic forms, we require in addition that the quadratic form should be rotationally symmetric. We prove that there are essentially two different choices: the square Laplacian and the quadratic variation. All the remaining possibilities are linear combinations, that is, form a vector space with these two as a basis.

To choose between the square Laplacian and the quadratic variation, we consider their respective Euler conditions and natural boundary conditions (Courant and Hilbert, 1953). The Euler conditions are identical, but the natural boundary conditions, which are derived from the statics of a deformed thin plate, favor the quadratic variation since they offer tighter constraint in this case.

2. Rotationally symmetric operators in vision

A major concern of Computer Vision is the isolation of constraints that combine with the information provided in the image to yield an interpretation. Early work on *polyhedra* (Clowes 1971, Huffman 1971, Mackworth 1973, Waltz 1972, Sugihara 1978, 1981, Kanade 1981) focussed upon the discovery of constraints deriving from the image forming process, constraints that relate image fragments, like junctions and lines, to their scene counterparts, vertices and edges. As Computer Vision turned its attention away from plane-faced objects to the natural world, other constraints were required. Often the constraints expressed some facet of the intuitive notion of "smoothness" and did so in a way that supported useful computations (Strat 1979, Brooks 1979, Ikeuchi and Horn 1981, Woodham 1978, Horn and Schunck 1981). Recently, smoothness and image forming have been combined using differential geometry (Grimson 1981, Witkin 1981, Binford 1981).

One constraint that is usually implicit, but is occasionally made explicit, expresses the idea that *perceptual* processes are often approximately *isotropic*. It seems that humans usually do not show strong directional preferences when detecting edges, motion, or reflectance boundaries. We seem to be equally adept at perceiving the layout and orientation of a visible surface regardless of its orientation relative to the view vector. Ullman(1976) argues for an *explicit isotropy* constraint in his work on subjective contours (see also Knuth 1979).

Processes that are isotropic are naturally computed by rotationally symmetric operators, since the values they return are unaffected by the coordinate system chosen for the image. Conversely, rotationally symmetric operators compute isotropic information. As we shall see, many operators that have been proposed for vision are not rotationally symmetric but directionally selective. Some authors have, however, proposed rotationally symmetric operators, particularly for early visual processing.

Precise definitions of rotational symmetry for functions, operators (or functionals), and, by specialization, matrices are given in the following section. In the rest of this section we assume that the definitions are already understood.

Some kinds of blurring in an image forming system can be approximated by convolution with a Gaussian. The rotationally symmetric Gaussian can be defined by:

$$G(r) = \frac{1}{2}\pi\sigma^2 \exp(\frac{-r^2}{2\sigma^2}).$$

Pratt(1978) presents several techniques, such as convolution with the generalized inverse of the blur function, for restoring the image. (see for example, his figures 14.2.1, 14.3.2).

The *Laplacian* $\Delta = f_{xx} + f_{yy}$ is well known to be rotationally symmetric[†] and its use has been proposed several times in Computer Vision and Image Processing. If an image is blurred in a way that can be approximately modelled by passing the image through a system with a Gaussian point spread function, then it can be sharpened by subtracting a multiple of its Laplacian (Rosenfeld and Kak 1976, p.184, Prewitt 1970, p. 107). Pratt(1978, figure 17.4.5) illustrates the use of the Laplacian for enhancing the edges in an image.

Weska, Dyer and Rosenfeld(1976) note that convolving a step edge with a Laplacian operator gives rise to a pulse pair: a negative pulse at the transition from the lower plateau to the edge, and a positive pulse at the transition from the edge to the upper plateau (see also Horn 1974, Marr and Hildreth 1980). They suggested that the image intensities at the locations of the positive and negative pulses could be used to set thresholds to use in segmenting the image into regions.

Several authors have noted the relative insensitivity of human perception to small intensity gradients (Herskovits and Binford 1970, Marr 1976, Marr and Hildreth 1980, McCann et. al. 1974). They have noted that the effect can be explained by assuming that the vision system uses operators approximating second derivatives. This so-called lateral inhibition effect seems to be performed by center surround operators in the retina (see for example Richter and Ullman 1980). The Laplacian is a rotationally symmetric second differential operator, and an attractive candidate to perform lateral inhibition.

The use of the Laplacian for edge detection was proposed by Horn(1974) in a study of the determination of lightness. Following Land and McCann(1971), Horn restricted attention to images of planes colored with patches of uniform reflectance or color. Within a patch, grey level variations are due to small variations in illumination, and they are smooth compared to the abrupt changes between patches. The conventional approach to detecting significant changes in intensity had been to note that the gradient of the image is small within a region, but is infinite across a reflectance boundary between regions. For a particular image tesselation and quantization of grey levels, the gradient is always finite. It is usually much larger, however, at a reflectance boundary than it is within a region. Horn(1974) rejected using the gradient since "the first partial derivatives are directional and thus unsuitable since they will for example completely eliminate evidence of edges running in a direction parallel to their direction of differentiation." The Laplacian is the lowest order linear combination of derivatives that is rotationally symmetric. A reflectance boundary can be *detected* by the paired positive and negative peaks on either side of the boundary, and *localized* by noting the position where the Laplacian crosses zero between the peaks[†].

[†] A proof of this is given in Section 3 below.

[†] See Binford(1981) for more on the distinction between detection and localisation of an intensity change.

Marr and Hildreth(1980) have proposed that edges are detected in the human visual system by an operator that approximates ΔG, where Δ is the Laplacian, and G is a rotationally symmetric Gaussian. We shall show in the next section that the application of a rotationally symmetric operator, such as the Laplacian, to a rotationally symmetric function, such as the Gaussian, is itself rotationally symmetric. It follows that the Marr-Hildreth operator is rotationally symmetric. Marr and Hildreth note that intensity changes occur at a number of scales and are often superimposed. They suggest that an image should be smoothed by a number of bandpass filters to isolate the changes at a particular range of scales. The Gaussian is chosen as the filter to optimize localization of changes in both the spatial and frequency domains.

We noted above that the Gaussian and the Laplacian have figured prominently in early visual processing. The Gaussian has mostly been used to approximate the point spread function corresponding to the blurring of a point source. Marr and Hildreth *deliberately* introduce Gaussian blurring. They further note that ΔG can be approximated by a difference of Gaussians, $G_1 - G_2$. Nishihara and Larson(1981) note that the difference of Gaussians is to be preferred on grounds of efficiency. Macleod(1972) proposes an edge detection operator that is the difference of two Gaussians. However, no analysis of its performance is given, and no indication is given that the operator approximated a low-pass filtered second derivative.

Regarding the use of the Laplacian, Marr and Hildreth do not seem to make isotropy an explicit constraint on edge detection. Instead, Hildreth(1980,page 13) notes that "a number of practical considerations, which will be illuminated in the discussion of the implementation, suggested that the ... operators not be directional". Suppose instead that directional operators are used. The simplest algorithm for edge detection has two stages. First, the image is convolved with the directional operators in "sufficiently many" directions. Second, the outputs are combined to determine the orientation and extent of intensity changes. Regarding the first stage, both Marr and Hildreth(1980, page 193) and Hildreth(1980, page 40) claim that the cost of convolving the image with a "sufficient" number of operators is excessive. They show that a single rotationally symmetric operator (the Laplacian) gives precisely the same results if a condition called "linear variation" holds. Regarding the second stage, Hildreth(1980, page 36) observes that edges in a direction close to that of the mask are elongated in the direction of the mask. She also notes that operators at several orientations give significant responses to any given edge, and that combining the responses is non-trivial.

There are two essentially different issues here that need to be clearly separated. Intensity changes first have to be detected and then localised as a set of "feature points" marking the position of the change in the image, and characteristics of the corresponding edge. The detection of feature points is inherently isotropic, as Horn(1974) noted. The feature points have then to be combined to produce *descriptions of edge segments*. Edge segments are clearly directional, indeed a central problem concerns the determination of the direction of an edge in an image. The computation of rich descriptions of edge segments is, as Hildreth notes, not at all easy. Marr's(1976) original Primal Sketch work was almost entirely concerned with it. Binford(1981) discusses the application of directional operators to compute the directionality of an edge.

The Gaussian and Laplacian are not the only rotationally symmetric operators that have been proposed in computer vision. Prewitt(1970, p. 107) observes that "derivatives of all orders can be used to form isotropic nonlinear differential operators, provided that derivatives of odd order appear only in even functions. The simplest of these ... is the squared gradient", namely $\nabla \cdot \nabla$, where ∇ is the column vector

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}.$$

Earlier in the same article, Prewitt(1970, p. 85) suggests that "the Hankel transformation enters naturally in the analysis of systems with isotropic point spread functions and greatly facilitates restoration." The suggestion does not appear to have been investigated in computer vision.

We noted earlier that an important aspect of modelling perception is the isolation of constraints which capture some facet of smoothness. Horn and Schunck(1981) consider the determination of optical flow fields and note that "if every point of the brightness pattern can move independently, there is little hope of recovering the velocities". One way to express the additional constraint of smoothness is to minimise the integral of the performance index

$$S(u,v) = (u_x^2 + u_y^2) + (v_x^2 + v_y^2),$$

where u and v are the x and y components of the optical flow, and subscripts denote partial differentiation. We shall show in the next section that this operator is rotationally symmetric. In many simple situations the smoothness constraint is violated significantly only at occluding boundaries.

We conclude this review of the use of rotationally symmetric operators in vision with Grimson's(1981) work on surface interpolation. As it will be the focus of Section 5, our remarks will be brief. The Marr-Poggio theory of human stereo vision yields the disparity (scaled depth) at matched edge points that are computed by the Marr-Hildreth approach described above. The disparity map is as sparse as the set of matched edge points, whereas human perception is of smooth surfaces passing through the given disparity points. Grimson (1981) interpolates a smooth surface from the given set of edge points by a local parallel algorithm that applies a rotationally symmetric operator to minimise the quadratic variation introduced above.

3. Conditions for rotational symmetry

A function $f:\Re^2 \mapsto \Re$ is *rotationally symmetric* if its polar form is only dependent on radial distance $r = (x^2+y^2)^{\frac{1}{2}}$ and not on direction $\phi = \tan^{-1}\frac{y}{x}$. Clearly, a function is rotationally symmetric if and only if it can be represented as a function of $(x^2+y^2)^{\frac{1}{2}}$ An alternative definition can be given that is often more convenient for functions, and that can be generalized to operators. A function is rotationally symmetric if and only if it yields the same value under an arbitrary rotation of coordinates.

An anticlockwise rotation from one set of image coordinates (x,y) to another (X,Y) is effected by a rotation matrix:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = R\begin{bmatrix} x \\ y \end{bmatrix}. \qquad (0)$$

For convenience, we shall denote $\cos\phi$ by c and $\sin\phi$ by s. To simplify notation, we shall not make explicit the dependence of the rotation matrix R on the angle ϕ.

A function f is rotationally symmetric if and only if the untransformed version $f(x,y)$ gives the same value as the transformed version $f(X,Y)$. We shall occasionally find it useful to borrow the mathematical shorthand that equates a function $f(X,Y)$ with a function of a single vector argument $f(R[x,y]^T)$.

Example 1. The function $f_1(x,y) = (x^2+y^2)$ is rotationally symmetric:

$$\begin{aligned} f_1(X,Y) &= ((xc+ys)^2 + (yc-xs)^2) \\ &= (x^2+y^2) \\ &= f_1(x,y). \end{aligned}$$

Example 2. The function $f_2(x,y) = xy$ is *not* rotationally symmetric:

$$\begin{aligned} f_2(X,Y) &= (xc+ys)(yc-xs) \\ &= xy\cos 2\phi + \frac{y^2-x^2}{2}\sin 2\phi, \end{aligned}$$

and so $f_2(X,Y) = f_2(x,y)$ only when $\phi = 0$ or $\phi = \pi$.

We can extend the definition of rotational symmetry to operators

$$O:(\Re^2 \mapsto \Re) \mapsto (\Re^2 \mapsto \Re).$$

An operator O is rotationally symmetric if $O(f)$ is a rotationally symmetric function, for all functions $f:\Re^2 \mapsto \Re$.

Example 3. The function produced by the operator O_1, defined by

$$O_1(f)(x,y) = e^{f(x,y)}$$

is rotationally symmetric if and only if f is. In general then, the operator O_1 is not rotationally symmetric. However, the Gaussian is rotationally symmetric, as it combines examples 1 and 3.

Most of the operators of interest in computer vision are combinations of the first and second directional derivatives $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial x\partial y}$, $\frac{\partial^2}{\partial y\partial x}$, and $\frac{\partial^2}{\partial y^2}$. We need to determine the effect of a coordinate rotation on these directional derivatives. By the chain rule,

$$\begin{aligned} \frac{\partial}{\partial x} &= \frac{\partial X}{\partial x}\frac{\partial}{\partial X} + \frac{\partial Y}{\partial x}\frac{\partial}{\partial Y} \\ &= c\frac{\partial}{\partial X} - s\frac{\partial}{\partial Y}. \end{aligned}$$

Similarly,

$$\frac{\partial}{\partial y} = s\frac{\partial}{\partial X} + c\frac{\partial}{\partial Y}$$

It follows that

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = R^T \begin{bmatrix} \frac{\partial}{\partial X} \\ \frac{\partial}{\partial Y} \end{bmatrix},$$

where T denotes matrix transpose. Since R is a rotation matrix, its transpose equals its inverse, so

$$\begin{bmatrix} \frac{\partial}{\partial X} \\ \frac{\partial}{\partial Y} \end{bmatrix} = R \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}. \qquad (1)$$

Operators in general, and differential operators in particular, depend upon the choice of coordinate frame. To make the dependence of the differential operator on the choice of coordinate frame explicit, we introduce the notation

$$O_{(x,y)}.$$

With this notation, equation (1) becomes

$$\nabla_{(X,Y)} = R\nabla_{(x,y)}, \qquad (2)$$

where $\nabla_{(x,y)}$ is the column vector

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}.$$

Proposition 1. Linear combinations of $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ are not rotationally symmetric.

Proof. Any linear form in the first directional derivatives has the form

$$[\lambda \quad \mu]\nabla_{(x,y)}.$$

The condition for rotational symmetry is

$$[\lambda \quad \mu]\nabla_{(X,Y)} = [\lambda \quad \mu]\nabla_{(x,y)}.$$

By equation (2),

$$[\lambda \quad \mu]\nabla_{(X,Y)} = [\lambda \quad \mu]R\nabla_{(x,y)},$$

and so the linear differential operator is rotationally symmetric if and only if

$$[\lambda \quad \mu] = [\lambda \quad \mu]R,$$

so that $[\lambda \quad \mu]$ is an eigenvector of R. The eigenvalues of R are $c+is$ and $c-is$. So there are no real eigenvectors unless ϕ is a multiple of π. Since the condition is not satisfied for all ϕ, no linear combination is rotationally symmetric. ∎

The same style of analysis can be applied to other combinations of first derivatives such as the operator

$$O_2(f) = \frac{f_x - f_y}{f_x + f_y}.$$

It is easy to show that $O_{2(X,Y)}$ is not equal to $O_{2(x,y)}$, for example when $\phi = \frac{\pi}{2}$.

In section 2, we referred to an operator proposed by Prewitt(1970), namely

$$\left(\frac{\partial}{\partial x}\right)^2 + \left(\frac{\partial}{\partial y}\right)^2,$$

that is, the vector dot product

$$\nabla^T_{(x,y)}\nabla_{(x,y)},$$

More generally, we often consider quadratic differential expressions such as

$$\nabla^T_{(x,y)}\begin{bmatrix}\lambda & \mu \\ \nu & \xi\end{bmatrix}\nabla_{(x,y)}.$$

Such an expression is called a *quadratic form* if the matrix is symmetric, that is $\mu = \nu$. By equation (1),

$$\nabla_{(X,Y)} = R\nabla_{(x,y)},$$

so that

$$\nabla^T_{(x,y)}M\nabla_{(x,y)} = \nabla^T_{(X,Y)}M\nabla_{(X,Y)},$$

if and only if

$$R^T M R = M,$$

where R is an arbitrary rotation matrix, and

$$M = \begin{bmatrix}\lambda & \mu \\ \nu & \xi\end{bmatrix}.$$

Since the transpose R^T of a rotation matrix R is the inverse of R, a quadratic form is rotationally symmetric if and only if the corresponding matrix M commutes with all rotation matrices. We will refer to matrices M having this property as being rotationally symmetric.

Lemma 1. A 2 by 2 matrix is rotationally symmetric if and only if it has the form

$$M = \begin{bmatrix}\lambda & \mu \\ -\mu & \lambda\end{bmatrix}.$$

Proof. We require $RM = MR$ for all rotation matrices R, that is

$$\begin{bmatrix}c & -s \\ s & c\end{bmatrix}\begin{bmatrix}\lambda & \mu \\ \nu & \xi\end{bmatrix} = \begin{bmatrix}\lambda & \mu \\ \nu & \xi\end{bmatrix}\begin{bmatrix}c & -s \\ s & c\end{bmatrix}.$$

Expanding, and equating terms, this holds if and only if

$$\mu + \nu = 0$$
$$\lambda = \xi.$$

Alternatively, only the operations of scaling by a constant k and multiplication by a rotation matrix R' commute with all rotation matrices in two dimensions So $M = kR'$ for some scale factor k and some rotation matrix R'. ∎

Proposition 2. Up to scaling, the only rotationally symmetric quadratic form in $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ is $\nabla_{(x,y)} \cdot \nabla_{(x,y)}$.

Proof. A quadratic form in $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ has the form

$$\nabla^T_{(x,y)}\begin{bmatrix}\lambda & \mu \\ \mu & \xi\end{bmatrix}\nabla_{(x,y)}. \tag{3}$$

To be rotationally symmetric, as well as symmetric (so that it is a quadratic form), Lemma 1 implies that

$$\lambda = \xi$$
$$\mu = 0.$$

It follows that the matrix in equation (3) is λI_2. ∎

The operator $f_x^2 + f_y^2$ is commonly used as a measure of the contrast across an intensity change. Notice that other measures of the contrast, such as $(f_x + f_y)^2$, $(f_x - f_y)^2$, or $\|f_x\| + \|f_y\|$ are not rotationally symmetric, and therefore respond differently to edges in different directions (see Rosenfeld and Kak 1976, p279).

We now consider linear and quadratic forms in $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial x \partial y}$ $\frac{\partial^2}{\partial y \partial x}$, and $\frac{\partial^2}{\partial y^2}$. It is convenient to *not* assume $\frac{\partial^2}{\partial x \partial y} = \frac{\partial^2}{\partial y \partial x}$ for the developments that follow.

The first task is to find a matrix R^* so that

$$\begin{bmatrix}\frac{\partial^2}{\partial x^2} \\ \frac{\partial^2}{\partial x \partial y} \\ \frac{\partial^2}{\partial y \partial x} \\ \frac{\partial^2}{\partial y^2}\end{bmatrix} = R^*\begin{bmatrix}\frac{\partial^2}{\partial X^2} \\ \frac{\partial^2}{\partial X \partial Y} \\ \frac{\partial^2}{\partial Y \partial X} \\ \frac{\partial^2}{\partial Y^2}\end{bmatrix}. \tag{4}$$

The (i,j) element of the matrix R[†] will be denoted by r_{ij}. Applying the chain rule as before, but this time to relate the second derivatives in (X,Y) to those in (x,y), we find that the four by four matrix R^* can be written in the form

[†] Recall the definition of the matrix R from equation (0).

$$R^* = \begin{bmatrix} r_{11}R^T & r_{21}R^T \\ r_{12}R^T & r_{22}R^T \end{bmatrix}. \qquad (5)$$

Definition 1. (ben Israel and Greville 1974, page 41)Let $A = [a_{ij}]$ and $B = [b_{ij}]$ be m by m and n by n matrices respectively. The mn by mn matrix $A \otimes B$, called the *Kronecker product* of A and B, is defined by multiplying each element $a(i,j)$ of A by the matrix B, to form the block matrix

$$\begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & \dots & a_{2m}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mm}B \end{bmatrix} \qquad (6)$$

With this notation,

$$R^* = R^T \otimes R^T,$$

so that

$$R^* = \begin{bmatrix} c^2 & -sc & -sc & s^2 \\ sc & c^2 & -s^2 & -sc \\ sc & -s^2 & c^2 & -sc \\ s^2 & sc & sc & c^2 \end{bmatrix}. \qquad (7)$$

Note that the elements of $A \otimes B$ are naturally indexed by 4-tuples:

$$\{A \otimes B\}_{ijkl} = a_{ij}b_{kl}.$$

We state without proof a number of simple properties of the $\otimes$ operation. They are essentially straightforward consequences of the properties of ordinary multiplication, and are stated without proof.

Proposition 3

(i) $(A \otimes B)^T = A^T \otimes B^T$

(ii) $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$

(iii) $(A \otimes B) \otimes C = A \otimes (B \otimes C)$

For the remainder of the paper, we restrict attention to the application of $\otimes$ to R and its transpose.

Proposition 4. The rotationally symmetric linear combinations of $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial x \partial y}$, $\frac{\partial^2}{\partial y \partial x}$, and $\frac{\partial^2}{\partial y^2}$ are linear combinations of the Laplacian $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, and the smoothness measure $\frac{\partial^2}{\partial x \partial y} - \frac{\partial^2}{\partial y \partial x}$.

Proof. Let the linear combination be

$$[\lambda \quad \mu \quad \nu \quad \xi] \begin{bmatrix} \frac{\partial^2}{\partial x^2} \\ \frac{\partial^2}{\partial x \partial y} \\ \frac{\partial^2}{\partial y \partial x} \\ \frac{\partial^2}{\partial y^2} \end{bmatrix}.$$

Following the proof of Proposition 1, the condition for rotational symmetry is

$$[\lambda \quad \mu \quad \nu \quad \xi] R^T \otimes R^T = [\lambda \quad \mu \quad \nu \quad \xi],$$

for all rotation matrices R and the corresponding rotation angle ϕ. Expanding $R^T \otimes R^T$ by equation (7), we find

$$[\lambda \quad \mu \quad \nu \quad \xi] \begin{bmatrix} c^2 & -sc & -sc & s^2 \\ sc & c^2 & -s^2 & -sc \\ sc & -s^2 & c^2 & -sc \\ s^2 & sc & sc & c^2 \end{bmatrix} = [\lambda \quad \mu \quad \nu \quad \xi],$$

so that

$$[\lambda \quad \mu \quad \nu \quad \xi] \begin{bmatrix} -s^2 & -sc & -sc & s^2 \\ sc & -s^2 & -s^2 & -sc \\ sc & -s^2 & -s^2 & -sc \\ s^2 & sc & sc & -s^2 \end{bmatrix} = [0 \quad 0 \quad 0 \quad 0].$$

It follows that

$$[\lambda - \xi \quad \mu + \nu \quad 0 \quad 0] \begin{bmatrix} -2s^2 & -2sc & -2sc & 2s^2 \\ 2sc & -2s^2 & -2s^2 & -2sc \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$
$$= [0 \quad 0 \quad 0 \quad 0]$$

The determinant of the upper left 2 by 2 submatrix is

$$(4s^4 + 4s^2c^2) = 4s^2.$$

Since this is not zero for all angles ϕ, $\lambda - \xi$ and $\mu + \nu$ are both zero. A basis for the infinite set of linear combinations satisfying these conditions is provided by setting λ and μ equal to one, which proves the Proposition. ∎

Before turning to quadratic forms, analogous to Proposition (2), we define a projection operator on $R^T \otimes R^T$ that makes explicit the assumption $f_{xy} = f_{yx}$.

Definition 2. Let $D = [d_{ij}]$ be a 4 by 4 matrix. The *projection* of D is the 3 by 3 matrix D^*:

$$\begin{bmatrix} d_{11} & (d_{12} + d_{13}) & d_{14} \\ (d_{21} + d_{31}) & (d_{22} + d_{32} + d_{23} + d_{33}) & (d_{24} + d_{34}) \\ d_{41} & (d_{42} + d_{43}) & d_{44} \end{bmatrix}$$

That is, the second and third columns as well as the second and third rows are combined by addition.

Proposition 5.

$$[a \quad b \quad b \quad c] D [a \quad b \quad b \quad c]^T$$

is equivalent to

$$[a \quad b \quad c] D^* [a \quad b \quad c]^T,$$

where D^* is the projection of D.

The proof is by equating terms, and is omitted. We now give the main result of this section.

Proposition 6. The rotationally symmetric quadratic forms in $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial x \partial y}$, $\frac{\partial^2}{\partial y \partial x}$, and $\frac{\partial^2}{\partial y^2}$ form a vector space. If $\frac{\partial^2}{\partial x \partial y} = \frac{\partial^2}{\partial y \partial x}$, the matrices associated with the rotationally symmetric quadratic forms project to 3 by 3 matrices of the form

$$\begin{bmatrix} \alpha+\beta & 0 & \beta \\ 0 & 2\alpha & 0 \\ \beta & 0 & \alpha+\beta \end{bmatrix}$$

It follows that the rotationally symmetric quadratic forms that satisfy $\frac{\partial^2}{\partial x \partial y} = \frac{\partial^2}{\partial y \partial x}$ form a vector space that has the quadratic variation, $(\frac{\partial^2}{\partial x^2})^2 + 2(\frac{\partial^2}{\partial x \partial y})^2 + (\frac{\partial^2}{\partial y^2})^2$, and the square Laplacian, $(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})^2$, as a basis.

Proof. Since the matrix in a quadratic form is defined to be symmetric, a quadratic form in $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial x \partial y}$, $\frac{\partial^2}{\partial y \partial x}$, and $\frac{\partial^2}{\partial y^2}$ can be written

$$\begin{bmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial x \partial y} & \frac{\partial^2}{\partial y \partial x} & \frac{\partial^2}{\partial y^2} \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} \frac{\partial^2}{\partial x^2} \\ \frac{\partial^2}{\partial x \partial y} \\ \frac{\partial^2}{\partial y \partial x} \\ \frac{\partial^2}{\partial y^2} \end{bmatrix},$$

where A and C are symmetric 2 by 2 matrices, and B is 2 by 2. As usual, the quadratic form is rotationally symmetric if and only if

$$R^T \otimes R^T \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} R^T \otimes R^T,$$

where R is an arbitrary rotation matrix. It follows that

$$\begin{bmatrix} cR^T & sR^T \\ -sR^T & cR^T \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} cR^T & sR^T \\ -sR^T & cR^T \end{bmatrix},$$

and hence that

$$\begin{bmatrix} cR^TA + sR^TB^T & cR^TB + sR^TC \\ -sR^TA + cR^TB^T & -sR^TB + cR^TC \end{bmatrix} = \begin{bmatrix} cAR^T - sBR^T & sAR^T + cBR^T \\ cB^TR^T - sCR^T & sB^TR^T + cCR^T \end{bmatrix}.$$

Equating submatrices, we find that for all rotation angles ϕ

$$c(R^TA - AR^T) + s(R^TB^T + BR^T) = 0, \tag{8}$$
$$c(R^TC - CR^T) - s(B^TR^T + R^TB) = 0, \tag{9}$$
$$s(CR^T - R^TA) + c(R^TB^T - B^TR^T) = 0, \tag{10}$$
$$s(R^TC - AR^T) + c(R^TB - BR^T) = 0. \tag{11}$$

Consider equation (10) or (11) when $\phi = \frac{\pi}{2}$. Equating terms, we find that

$$\begin{aligned} a_{11} &= c_{22} \\ a_{22} &= c_{11} \\ a_{12} &= -c_{21} \\ a_{21} &= -c_{12}. \end{aligned} \tag{12}$$

Similarly, equation (8) or (9) when $\phi = \frac{\pi}{2}$ yields

$$b_{11} + b_{22} = 0. \tag{13}$$

Expanding equation (8) for general ϕ yields

$$b_{11} + a_{12} = 0, \tag{14}$$
$$b_{22} - a_{21} = 0, \tag{15}$$
$$b_{21} + b_{12} + a_{22} - a_{11} = 0. \tag{16}$$

Combining equations (12) through (16) we find that in order to be rotationally symmetric, the matrix

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

has the form

$$\begin{bmatrix} \alpha+\beta & \gamma & -\gamma & \beta \\ \gamma & \alpha-\delta & \delta & \gamma \\ -\gamma & \delta & \alpha-\delta & -\gamma \\ \beta & \gamma & -\gamma & \alpha+\beta \end{bmatrix}.$$

A matrix of this form projects to

$$\begin{bmatrix} \alpha+\beta & 0 & \beta \\ 0 & 2\alpha & 0 \\ \beta & 0 & \alpha+\beta \end{bmatrix},$$

where $\alpha = b_{12} - a_{11}$ and $\beta = b_{12}$. It is easy to show that linear combinations of matrices of this form are of the same form, so that the rotationally symmetric quadratic forms constitute a vector space. Clearly, the square Laplacian and the quadratic variation, corresponding to the cases $\alpha = 1, \beta = 0$ and $\alpha = 0, \beta = 1$ respectively, form a basis.∎

We show that the measure of smoothness of optical flow proposed by Horn and Schunck(1981) is rotationally symmetric. Recall from section 2 that the measure is defined by the operator

$$S(u,v) = (u_x^2 + u_y^2) + (v_x^2 + v_y^2).$$

We extend the Kronecker product operator $\otimes$ to vectors, and then show how to define $S(u,v)$ in terms of vector Kronecker products.

Definition 3. (a) Let $\underline{a} = [a_1 \ldots a_m]$ and $\underline{b} = [b_1 \ldots b_n]$ be vectors. The *Kronecker product* of $\underline{a}$ and $\underline{b}$ is the mn dimensional vector $[a_1b_1 \ldots a_1b_n \, a_2b_1 \ldots a_mb_n]$,

(b) By extension, if $\underline{O} = [O_1 \ldots O_m]$ is a vector of operators and $\underline{f} = [f_1 \ldots f_n]$ is a vector of functions, the Kronecker product of $\underline{O}$ and $\underline{f}$ is the mn dimensional vector of functions

$$[O_1(f_1) \ldots O_1(f_n) \ldots O_m(f_n)].$$

The components u and v of optical flow are functions of x, y, and t. Recall that $\nabla_{(x,y)} = [\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y}]^T$. According to definition 3,

$$\nabla_{(x,y)} \otimes [u \quad v]^T = [\frac{\partial u}{\partial x} \quad \frac{\partial u}{\partial y} \quad \frac{\partial v}{\partial x} \quad \frac{\partial v}{\partial y}],$$

so that

$$S(u,v) = (\nabla_{(x,y)} \otimes [u \quad v]^T) \cdot (\nabla_{(x,y)} \otimes [u \quad v]^T).$$

If the coordinate frame is rotated through ϕ by the matrix R, the optical flow components become $R[u \quad v]^T$. The Horn-Schunck measure is rotationally symmetric if and only if

$$(R \otimes R)^T (R \otimes R) = I_4,$$

where I_4 is the 4 by 4 identity matrix. The rotational symmetry is a simple consequence of Proposition 3.

A rotationally symmetric operator has the general form

$$O_{(x,y)}(\nabla, \nabla \otimes \nabla, \nabla \otimes \nabla \otimes \nabla, \cdots),$$

and its application to a rotationally symmetric function $f(x, y)$ has the form

$$O_{(x,y)}(f(x, y)).$$

To see that this is rotationally symmetric, we rotate the coordinate frame to (X, Y) by a matrix R as before. Since O and f are rotationally symmetric, all the occurences of R (including its Kronecker square, cube, and so on) introduced by the frame change can be deleted. It follows that the application of a rotationally symmetric operator to a rotationally symmetric function is itself rotationally symmetric. In particular, the $\Delta(G)$ filters of the Marr-Hildreth theory of edge detection are rotationally symmetric.

4. Vision as a conservative process

The second theme of this paper is that a number of vision modules construct the *most conservative* interpretation that is consistent with the given data, and that is subject to an appropriate set of suitably formulated constraints. A major concern of Computer Vision has always been the isolation of constraints that enable the interpretation of an image. Constraints embody observations about the way the world is, at least, most of the time. Although such observations can be as specific as cataloging familiar figures and shapes, it has proved more fruitful to first uncover constraints that correspond to general observations that are widely applicable. Constraints are used together with the data computed from the image to construct an interpretation. The *representations* of the information from the image and the constraints determine, and are determined by, the interpretation process. For example, early blocks world programs represented constraints as catalogs of labellings, an approach that led naturally to search processes for interpretation (Clowes 1971, Kanade 1981).

As Computer Vision has turned its attention to images of the natural world, constraints have concerned the smoothness of surfaces and movement. The relationship to boundary value problems of physics and mathematics suggests itself. The information computed from the image sets the boundary conditions, and the constraints determine which (and whether a) solution to the boundary value problem is found. Horn(1974) solved an instance of Poisson's problem using Green's functions to determine the lightness of an image.

Following a different approach, Ullman(1979a) studied the perception of apparent motion generated by two successive frames consisting of isolated dots of equal intensity moving independently of each other. Without constraint, all possible pairings, or "correspondences", of dots in the first frame with dots in the second are equally likely. Ullman defined the "most likely" correspondence to be the one that minimized the sum

$$\sum_{\substack{1 \le i \le n \\ 1 \le j \le m}} x_{ij} q_{ij},$$

where n is the number of dots in the first frame, m is the number of dots in the second frame, and x_{ij} is one if the ith dot of the first frame P_i is paired with the jth dot of the second frame Q_j, else zero. The weight q_{ij} is the "cost" of pairing P_i with Q_j, and might, for example, be related to the image distance between the paired points. The problem of finding the minimal correspondence is considered in terms of integer programming. If correspondences are assumed to be covering mappings, the following linear constraints apply to the x_{ij}:

$$\forall i, 1 \le i \le n \sum_{1 \le j \le m} x_{ij} \ge 1,$$

and

$$\forall j, 1 \le j \le m \sum_{1 \le i \le n} x_{ij} \ge 1.$$

Ullman restricted the set of Q_j that can be paired with P_i to those whose positions were close to P_i. Following Arrow, Hurwicz, and Uzawa(1958), he set up the iterative scheme

$$\begin{aligned} x_{ij}^{t+1} &= u_i^t + v_j^t - q_{ij}^t \\ u_i^{t+1} &= \sum_{1 \le i \le n} x_{ij}^t - 1 \\ v_j^{t+1} &= \sum_{1 \le j \le m} x_{ij}^t - 1 \end{aligned}$$

The approach can be extended to mappings that are not one-one, as well as to continous motion. A major problem with the approach is guaranteeing the convergence of the algorithm. This is determined largely by the properties of the costs q_{ij}, but this was not investigated, aside from a comment on the empirical determination of the q_{ij} (see also Ullman 1979b).

One limitation of Ullman's approach is that it is restricted to minimizing a *known* linear objective function that is subject to linear constraints. The method can be extended to constrained nonlinear programming in which the goal is to minimize a known function $f(\underline{x})$ subject to a set of equality and inequality constraints of the form $g_i(\underline{x}) \leq 0$. In general, however, criteria based on other than intuition need to be found for selecting the function f to be minimized. To do this, one can apply the calculus of variations (see for example Courant and Hilbert 1953, chapter IV). The familiar differential shows how to find a real valued parameter that minimizes some function. The calculus of variation extends the differential calculus by showing how one can determine a *function* f^*, which is subject to a given set of boundary conditions, and minimizes the integral

$$\mathcal{F}(f) = \int \int_G F(x, y, f, f_x, f_y, f_{xx}, f_{xy}, f_{yy}) dx dy \qquad (17)$$

over a given region of integration G[†]. The function F is called a "*performance index*" and generalizes the notion of cost function associated with linear and nonlinear programming. In the next section we shall consider the choice of a performance index for interpolating smooth surfaces from one-dimensional boundary conditions.

Associated with a variational problem of the form (17) is the *Euler equation*, which provides a necessary, though by no means sufficient, condition which a function f must satisfy if it is to minimize the variational integral $\mathcal{F}(f)$. For the particular variational problem given in equation (17), the Euler equation is

$$F_f - \frac{\partial}{\partial x} F_{f_x} - \frac{\partial}{\partial y} F_{f_y} + \frac{\partial^2}{\partial x^2} F_{f_{xx}} + \frac{\partial^2}{\partial x \partial y} F_{f_{xy}} + \frac{\partial^2}{\partial y^2} F_{f_{yy}} = 0. \qquad (18)$$

In the case that there is only a single dependent variable x, the partial derivatives are total and the Euler equation becomes

$$F_f - \frac{d}{dx} F_{f_x} + \frac{d^2}{dx^2} F_{f_{xx}} = 0. \qquad (19)$$

There are two important considerations associated with the use of the calculus of variations. First, unlike the differential calculus, the existence of an extremum f^* of the integral given in equation (17) cannot be taken for granted. Courant and Hilbert(1953, p. 173) note that "a characteristic difficulty of the calculus of variations is that problems which can be meaningfully formulated may not have solutions". Conditions for the existence of a minimum have recently been proposed by Grimson(1981) and will be discussed in the next section.

Second, associated with any variational problem is a set of *natural boundary conditions* which imposes a necessary condition on any feasible solution to the Euler equation at the boundary. Courant and Hilbert(1953, p. 211) note that "in general, we can, by adding boundary terms or boundary integrals essentially modify the natural boundary conditions without altering the Euler equations". Determining the "most conservative" solution means finding a performance index that guarantees the existence of an extremum function f^* and provides the tightest set of natural boundary conditions that are consistent with the given data.

The calculus of variations has recently been applied by a number of authors to interpolate plane and space curves and surfaces. We review the applications in that order. First, Horn(1981) has recently determined the curve which passes through two specified points with specified orientation while minimizing

$$\int \kappa^2 ds, \qquad (20)$$

where κ is the curvature and s is the arc length. This is the true shape of a spline used in "lofting" (Faux and Pratt 1979, p. 228). In a thin beam, curvature is proportional to the bending moment. The total elastic energy stored in a thin beam is therefore proportional to the integral of the square of the curvature. Since the shape taken on by a thin beam is the one which minimizes the internal strain energy, the curve that solves equation (20) is called the "curve of least energy". The variational problem is to minimize

$$\int \frac{f_{xx}^2}{(1 + f_x^2)^{\frac{5}{2}}} dx.$$

This has the form of equation (17). Horn(1981, page 19) shows that the Euler equation is

$$-c\kappa = \sqrt{\cos \psi},$$

where ψ is the angle between the tangent to the curve and the axis of symmetry. The solution to this differential equation is an incomplete elliptic integral of the first kind. Brady, Grimson, and Langridge(1980) consider a "small angle" approximation to the curve of least energy, in which first derivatives can be ignored. The performance index that they use is f_{xx}^2, for reasons that will become evident in the next section. They find that in that case the solution is a cubic. Horn(1981, page 2) notes that the fact that a curve has near minimum energy does not mean that it lies close to the curve of minimum energy. Note that the existence of the curve of least energy is guaranteed as Horn has derived an analytical formula for it. Approximations to it, such as Brady, Grimson, and Langridge's are similarly guaranteed to exist.

Barrow and Tenenbaum(1981) investigate the problem of interpreting a line as the image of a space curve that is the occluding edge of an object. They observe that the problem has two parts: (i) determining the tangent vector $\underline{t}$ at each point on the space curve, and (ii) determining the surface normal at each point, given that it is constrained to be orthogonal to the tangent. They suggest minimizing a performance index F that is a function of the curvature κ

[†] For simplicity of presentation, we restrict attention to functions f of one or two variables x, y.

and the torsion τ (possibly together with their derivatives), and expresses a suitable notion of "smoothness". They first consider uniformity of curvature as a measure of smoothness, that is $F = \frac{ds}{ds} = \kappa_1$, where s measures distance along the space curve. They reject this measure on the grounds that κ_1 can be made arbitrarily small by "stretching out the space curve so that it approaches a twisting straight line". To overcome this difficulty, they propose that the space curve should also be "as planar as possible or, more precisely, that the integral of its torsion should be minimized".

Barrow and Tenenbaum finally suggest finding the space curve that projects to the given image line and minimizes the performance index $[\frac{d(\kappa \underline{b})}{ds}]^2$, where $\underline{b}$ is the binormal. They report that an algorithm based on their analysis produced the "correct 3-D interpretations for simple and closed curves, such as an ellipse, which was interpreted as a circle". However, they note that the rate of convergence was slow and dependent on the initial data. No consideration is given to the Euler equations, to the existence of an extremum given a line drawing $\{x(s), y(s)\}$, or to the natural boundary conditions associated with the performance index $[\frac{d(\kappa b)}{ds}]^2$. Empirical evidence that the method works on a number of simple test cases is encouraging; but there is no analysis of the scope of the method.

In the same paper, Barrow and Tenenbaum(1981) consider the interpolation of a smooth surface from depth and local surface orientation values at all points along the surface boundary. Their approach is to "seek a technique that yields exact reconstructions for the special symmetric cases of spherical and cylindrical surfaces, as well as intuitively reasonable reconstructions for other smooth surfaces." (Barrow and Tenenbaum 1981). They observe that if $\underline{n}$ is the surface normal of a cylinder, then the x and y components of the normal $\underline{n}_x$ and $\underline{n}_y$ are linear functions of x and y, so long as the axis of the cylinder lies in the xy-plane. This observation forms the basis of an algorithm to estimate the surface normal by least squares fitting of the parameters of the partial derivatives of the normal. As before, no analysis is given of the Euler equation, the natural boundary conditions, nor the convergence of their algorithm for different types of surface.

5. A performance index for surface interpolation.

In the review of the application of the calculus of variations to visual perception in the previous section we drew attention to three important considerations. First, the Euler equations provide a necessary condition on possible extremal functions. Second, the existence of an extremum cannot be taken for granted, even when the minimization problem seems plausible on some grounds. Third, the natural boundary conditions impose a necessary condition on any feasible solution to the Euler equation at the boundary. The most thorough analysis of the second of these problems in Computer Vision, framed in the context of surface interpolation, is due to Grimson(1981), who proves the following theorem.

Theorem (Grimson, see Rudin(1973)) Suppose there exists a complete semi-norm F on a space of functions $\mathcal{F}$, and that F satisfies the parallelogram law. Then, every nonempty closed convex set $\mathcal{E} \subseteq \mathcal{F}$ contains a unique element f^* of minimal norm $F(f^*)$, up to possibly an element of the null space of F.

A semi-norm F is a function $V \mapsto \Re^+$ from a vector space V to the positive real numbers that satisfies

$$F(v + w) \leq F(v) + F(w)$$
$$F(\alpha v) = |\alpha| F(v).$$

Informally, a semi-norm is a generalization of the Euclidean metric, and provides a measure of a vector. The first condition generalizes the triangle inequality, for example. The *null space* of the semi-norm F consists of all those vectors v_0 that map to zero. Since

$$F(v + v_0) = F(v),$$

any element of the null space can be added to a vector of minimal norm to yield another vector of minimal norm. Hence the qualifying phrase "unique ... up to possibly an element of the null space of F". The parallelogram law states that

$$[F(v + w)]^2 + [F(v - w)]^2 = 2[F(v)]^2 + 2[F(w)]^2,$$

for all vectors v, w. Finally, the semi-norm is complete if all Cauchy sequences converge. As is well known, the elements of vector spaces can be functions. This enables Grimson to prove the following Corollary, that guarantees the existence of an extremum function in calculus of variations "most conservative" interpolation problems.

Corollary (Grimson 1981). Let the set of known points be $\{(x_i, y_i, z_i) \mid 1 \leq i \leq n\}$. Let $\mathcal{F}$ be a vector space of possible functions on $\Re^2$ and let $\mathcal{E}$ be the subset of $\mathcal{F}$ that interpolates the known data. That is, for all functions $f \epsilon \mathcal{E}, f(x_i, y_i) = z_i$. Let F be a complete semi-norm on $\mathcal{E}$ that satisfies the parallelogram law. Then there exists a unique (up to the null space of F) function f^* that interpolates the data and has minimal norm. In particular, if F is a performance index then there is a function f^* that minimizes the integral

$$\mathcal{F}(f) = \int F.$$

In short, if the conditions of the Corollary are fulfilled, the existence of a "most conservative" surface that meets the boundary conditions is guaranteed. As we shall see, the condition of being a semi-norm is the most restrictive required of the performance index. The conditions are sufficient to guarantee the existence of a minimum, but they are not necessary. For example, κ^2 is not a seminorm[†]; nevertheless Horn's(1981) analysis shows that there is a unique minimum. It is far from clear whether Barrow and Tenenbaum's(1981) analyses of curve and surface interpolation have a guaranteed minimum in all cases.

Grimson notes that several intuitively plausible performance indices are not semi-norms. For example, the two most popular measures of curvature are not. Suppose that κ_1 and κ_2 are the principal curvatures of a surface(Faux and Pratt 1979, p. 111), then the Gaussian curvature κ_g is the product $\kappa_1\kappa_2$ and the mean curvature κ_m is the sum $\kappa_1 + \kappa_2$. For a surface $f(x,y)$,

$$\kappa_g(f) = \frac{f_{xx}f_{yy} - f_{xy}^2}{(1 + f_x^2 + f_y^2)^2}.$$

Since the curvatures can be negative, while a semi-norm is required to be positive, it is necessary to investigate

$$\left\{\iint \kappa_g^2 dx dy\right\}^{\frac{1}{2}}.$$

Grimson(1981) observes that $\kappa_g^2(\alpha f) \neq |\alpha|\kappa_g^2(f)$ because of the denominator. If f_x and f_y are small, the denominator is approximately equal to one, and so the expression is approximately equal to the numerator. Note that it is

$$f_{xx}f_{yy} - f_{xy}^2, \tag{21}$$

Grimson shows that the mean curvature κ_m is also not a semi-norm for exactly the same reason. The analogous small angle approximation is

$$(f_{xx} + f_{yy})^2 = (\Delta f)^2,$$

the square Laplacian, which is a semi-norm. We find it convenient to denote the square Laplacian by F_l. Grimson(1981) chooses the quadratic variation

$$f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2,$$

on the grounds that its null space, consisting of all linear functions, is smaller than the null space of the square Laplacian. If we denote the quadratic variation by F_q, we see that the approximation to the Gaussian curvature given in equation (21) is $\frac{(F_l - F_q)}{2}$.

How shall we choose a performance index for surface interpolation, given that it has to satisfy the conditions of the Corollary? We have exhibited three candidates, are there more? Notice first that each of the semi-norms given above are quadratic forms in f_{xx}, f_{xy}, and f_{yy}. It is easy to show that any quadratic form satisfies the semi-norm and parallelogram conditions, and so there is an infinite set of plausible semi-norms to use to find the "most conservative" interpolated surface. We need an extra condition, and the one we choose is rotational symmetry, since we suppose that surface interpolation is an isotropic process. Proposition 6 of section 3 shows that the rotationally symmetric quadratic forms in f_{xx}, f_{xy}, and f_{yy} form a vector space that has the square Laplacian and the quadratic variation as a basis. The choice of which performance index to use is thus effectively reduced to the square Laplacian, the quadratic variation, and linear combinations of them. How shall we choose between those two? In the light of our earlier discussion, two criteria suggest themselves: the Euler equations and the natural boundary conditions.

Proposition 7. All rotationally symmetric quadratic forms lead to an identical Euler equation

$$\Delta^2(f) = 0.$$

Proof. We exploit the fact that the square Laplacian and the quadratic variation are a basis of the rotationally symmetric quadratic forms.

a.*Square Laplacian*: The performance index is

$$F_l = (f_{xx} + f_{yy})^2.$$

By equation (18) the Euler equation is

$$\frac{\partial^2}{\partial x^2}\{2(f_{xx} + f_{yy})\} + \frac{\partial^2}{\partial y^2}\{2(f_{xx} + f_{yy})\} = 0,$$

that is

$$(\Delta f)^2 = 0,$$

as required.

b.*Quadratic variation*: The Euler equation is

$$2f_{xxxx} + 4f_{xyxy} + 2f_{yyyy} = 0,$$

that is

$$(\Delta f)^2 = 0,$$

provided that f is continuous of fourth order.

c.*Linear combinations of F_l and F_q*.: Linear combinations clearly give rise to the identical Euler equation.∎

The gist of Proposition 7 is that there is no difference between F_q and F_l in the interior, away from the boundary. We can see the result of Proposition 7 in an alternative interesting way. Recall that

$$F_l - F_q = 2(f_{xx}f_{yy} - f_{xy}^2),$$

is the semi-norm approximation to the Gaussian curvature (equation (21)). The latter expression is an instance of a *divergence expression*, and Courant and Hilbert(1953, p. 196) note "If the difference between the integrands of two variational problems is a divergence expression, then the Euler equations and therefore the families of extremals are identical for the two variational problems."

[†] Which is why Brady, Grimson, and Langridge(1980) used the small angle approximation f_{xx}^2.

Since F_q and F_l have identical Euler equations, we analyze their natural boundary conditions in order to choose between them. We could approach this problem directly; but a more revealing route is available. Courant and Hilbert(1953, p250) consider the statics of a thin plate. In particular they determine the shape it assumes for a given force $p(s)$ along its boundary Γ and bending moment $m(s)$ normal to its boundary.

Courant and Hilbert note that the energy stored in the plate is the integral of a quadratic form in the principal curvatures κ_1 and κ_2 of the surface, a result which can be derived by noting that the elastic energy stored in a thin strip (corresponding to any normal section) is proportional to the square curvature. It follows that the stored energy is locally

$$\begin{aligned}\mathcal{E}_1 &= \alpha(\kappa_1^2 + \kappa_2^2) + 2\beta\kappa_1\kappa_1 \\ &= \alpha(\kappa_1 + \kappa_2)^2 + 2(\beta - \alpha)\kappa_1\kappa_1 \\ &= \alpha\kappa_m + 2(\beta - \alpha)\kappa_g\end{aligned}$$

If we assume that the first derivatives are small, we can use the same approximation to the curvature used in equation (21):

$$\begin{aligned}&\sim \alpha F_l + 2(\beta - \alpha)\frac{(F_l - F_q)}{2} \\ &= \beta F_l + (\alpha - \beta)F_q \\ &= \alpha(\mu F_l + (1 - \mu)F_q),\end{aligned}$$

where $\mu = \frac{\beta}{\alpha}$. It follows that the energy stored in the thin plate is a convex linear combination of the square Laplacian and the quadratic variation, which formally establishes its connection to the visual perceptual problem studied here. Observe that setting the weight $\mu = 1$ gives the square Laplacian, while setting it equal to zero gives the quadratic variation.

Energy is also associated with the external force and moment applied at the boundary. Let the force per unit length be $p(s)$ along the boundary Γ of the plate and a bending moment $m(s)$ applied normal to the plate. Courant and Hilbert(1953, p. 251) show that the natural boundary conditions associated with the plate are

$$\begin{aligned}p(s) &= -\Delta f + (1 - \mu)(f_{xx}x_s^2 + 2f_{xy}x_sy_s + f_{yy}y_s^2) \\ m(s) &= \frac{\partial}{\partial n}\Delta f \\ &\quad + (1 - \mu)\frac{\partial}{\partial s}(f_{xx}x_sx_n + f_{xy}(x_sy_n + x_ny_s) + f_{yy}y_sy_n),\end{aligned}$$

where x_s and x_n are the partial derivatives of x in the directions of the tangent and normal to the boundary of the thin plate. Similar for y_s and y_s. That is,

$$\begin{aligned}-\Delta f + (1 - \mu)([x_sy_s]H[x_sy_s]^T) &= p(s) \\ \frac{\partial}{\partial n}\Delta f + (1 - \mu)\frac{\partial}{\partial s}([x_ny_n]\,H\,[x_sy_s]^T) &= m(s),\end{aligned}$$

where H is the Hessian matrix

$$\begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}.$$

Gladwell and Wait(1979) quote version of this result due to Agmon(1965), that the biharmonic operator, which we showed was the natural boundary condition for the surface interpolation problem, has Dirichlet forms that are linear combinations of the square Laplacian and the quadratic variation. As an example of the constraint, consider a straight line contour aligned with the x-axis. Then $[x_sy_s] = [1\ 0]$ and $[x_n\ y_n] = [0\ 1]$. The natural boundary conditions reduce to

$$\begin{aligned}f_{yy} + \mu f_{xx} &= -p(s) \\ f_{yyy} + (2 - \mu)f_{yxx} &= m(s).\end{aligned}$$

The constraint is tightest when μ is not equal to one. A similar result can be obtained for a straight line contour inclined at an angle α to the x-axis. The first of the natural boundary conditions is

$$\begin{aligned}f_{xx}(\sin^2\alpha + \mu\cos^2\alpha) + f_{yy}(\cos^2\alpha + \mu\sin^2\alpha) \\ + (1 - \mu)\sin 2\alpha f_{xy}\end{aligned}$$

If $\mu = 1$, there is no constraint from the cross derivative. If μ is not equal to 1, at most one of the terms can be zero. We conclude that interpolation problems in which the small angle approximations used throughout our analysis hold it is preferable to choose μ not equal to one, that is to say to not use the square Laplacian as a performance index. The quadratic variation is an obvious choice, but so are linear combinations of the square Laplacian and the quadratic variation for which μ is not equal to one. Grimson(1981) chooses the quadratic variation since its null space is smaller than that of the square Laplacian. This is a precise way of saying that it imposes a tighter constraint. For example, the function $f(x, y) = xy$ is in the null space of the square Laplacian but not in the null space of the quadratic variation. Since the quadratic variation has the smallest null space among the linear combinations of the square Laplacian and quadratic variation, all linear combinations have the same null-space ? only the Laplacian itself is different... this is an additional reason for choosing it. We would further expect that any differences between the quadratic variation and the square Laplacian would show up near the given boundary data but not in the interior, far removed from the boundary. This is what Grimson(1981) finds in a set of examples that compare surfaces interpolated using the quadratic variation and the square Laplacian.

6. Acknowledgement

The authors thank Eric Grimson for comments on drafts of this article.

7. References

Agmon S.. *Lectures on elliptic boundary value problems* , Van Nostrand, New Jersey, 1965.

Arrow K. J., Hurwicz L., and Uzawa H. *Studies in linear and nonlinear programming* , Stanford University press, Stanford, 1958.

Barrow H. G. and Tenenbaum J. M. "Interpreting line drawings as three dimensional surfaces," *Artificial Intelligence* 17 (1981), 75-116.

ben Israel A. and Greville T. N. *Generalized inverses: theory and applications* , John Wiley and sons, New York, 1974.

Binford T. O. "Inferring surfaces from images," *Artificial Intelligence* 17 (1981).

Brady Michael, Grimson W. E. L., and Langridge D. "The shape of subjective contours," *Proc. AAAI* 1 (1980), 15-17.

Brooks M. J. "Surface normals from closed paths," *Proc. Int. Jt. Conf. Artificial Intelligence* 6 (1979), 98-101.

Clowes M. B. "On seeing things," *Artificial Intelligence* 2 (1971), 79-112.

Courant R., and Hilbert D. *Methods of mathematical physics* , John Wiley Interscience, New York, 1953.

Faux I. D. and Pratt M. J. *Computational Geometry for design and manufacture* , Ellis Horwood, Chichester UK, 1979.

Gladwell I. and Wait. R. (eds.). *Survey of numerical methods for partial differential equations* , Clarendon Press, Oxford, 1979.

Grimson, W. E. L. *From images to surfaces: a computational study of the human early visual system* , MIT Press, Cambridge, 1981.

Herskovits A and Binford T. O. "On Boundary Detection," MIT, AI Memo 183, 1970.

Hildreth E. C. Implementation of a theory of edge detection (MS dissertation), also AI-TR 579, MIT, 1980.

Horn B. K. P. "Determining lightness from an image," *Computer Graphics and Image Processing* 3 (1974), 277-299.

Horn B. K. P. "The curve of least energy," MIT, AI Memo 610, 1981.

Horn B. K. P. and Schunck B. G. "Determining optical flow," *Artificial Intelligence* 17 (1981), 185-203.

Huffman D. A. "Impossible Objects as Nonsense Sentences," *Machine Intelligence 6*, eds. Meltzer B. and Michie D., Edinburgh University Press, 1971.

Ikeuchi K. and Horn B. K. P. "Numerical shape from shading and occluding boundaries," *Artificial Intelligence* 17 (1981), 141-184.

Kanade T. "Recovery of the three dimensional shape of an object from a single view," *Artificial Intelligence* 17 (1981), 409-460.

Knuth D. E. "Mathematical typography," *Bull. Amer. Math. Soc. (new series)* 1 (1979), 337-372.

Land E. H. and McCann J. J. "Lightness and Retinex Theory," *J. Optical Society of America* 61 (1971), 1-11.

Mackworth A. K. "Interpreting Pictures of Polyhedral Scenes," *Artificial Intelligence* 4 (1973), 121-137.

Macleod I. D. G. "Comments on "Techniques for edge detection"," *Proc. IEEE* **60, part 3** (1972), 344.

Marr D. "Early Processing of visual information," *Phil. Trans. R. Soc. Lond. B* 275 (1976), 483-524.

Marr D. and Hildreth E. "Theory of edge detection," *Proc. R. Soc. Lond. B* 207 (1980), 187-217.

McCann J. J., Savoy R. L., Hall J. A. Jr., and Scarpetti J. J. "Visibility of continuous luminance gradients," *Vision Research* 14 (1974), 917-927.

Nishihara H. K. and Larson N. G. "Toward a real time implementation of the Marr-Poggio stereo matcher," *Proceedings of the Image Understanding Workshop* eds. Lee Baumann, 1981.

Pratt William K. *Digital image Processing* , Wiley Interscience, New York, 1978.

Prewitt J. M. S. "Object enhancement and extraction," *Picture Processing and psychopictorics* eds. Lipkin R. S., and Rosenfeld Azriel, 75-149, 1970.

Richter J. and Ullman S. "A model for the spatio-temporal organization of X and Y-type Ganglion cells in the primate retina," MIT, AI Memo 573, 1980.

Rosenfeld Azriel and Kak Avinash C. *Digital Picture Processing* , Academic, New York, 1976.

Rudin W. *Functional Analysis* , McGraw-Hill, New York, 1973.

Strat T. M. A numerical method for shape from shading from a single image, MIT, 1979.

Sugihara K. "Picture language for skeletal polyhedra," *Comp. Graphics and Image Proc* 8 (1978), 382-405.

Sugihara K. "Algebraic and combinatorial approach to the analysis of line drawings of polyhedra," Electrotechnical Lab., Japan, RMI 81-02, 1981.

Ullman S. "Filling in the gaps: the shape of subjective contours and a model for their generation," *Biol. Cyb* 25 (1976), 1-6.

Ullman S. *The interpretation of visual motion* , MIT Press, Cambridge, Mass, 1979a.

Ullman, S. "Relaxation and constrained optimisation by local processes," *Computer Graphics and Image Processing* 10 (1979b), 115-125.

Waltz D. L. Generating semantic descriptions from drawings of scenes with shadows, MIT, 1972.

Weska J., Dyer C., and Rosenfeld Azriel. "A comparative study of texture measures for terrain classification," *IEEE Trans. Syst. Man and Cyb* **SMC-6** (1976), 269-285.

Witkin Andrew P. "Recovering surface shape and orientation from texture," *Artificial Intelligence* **17** (1981), 17-47.

Woodham R. J. Reflectance map techniques for analyzing surface defects in metal castings, MIT, 1978.

AD-P000 123

SEGMENTATION AND AGGREGATION: AN APPROACH TO FIGURE-GROUND PHENOMENA

David G. Lowe and Thomas O. Binford

Computer Science Department
Stanford University, Stanford, California 94305

Abstract

We describe a new approach to low-level vision in which the task of image segmentation is to distinguish meaningful relationships between image elements from a background distribution of random alignments. Unlike most previous approaches, which start from idealized models of what we wish to detect in the world, this approach is not based on prior world knowledge and uses measurements which can be computed directly from the input signal. Groupings of image elements are formed over a wide range of sizes and classes while attempting to make use of all available statistical information at each level of the grouping hierarchy, resulting in far more sensitive discrimination than is possible from just local measurements. This paper explores the range of grouping capabilities and discriminations exhibited by the human visual system and discusses the application of the meaningfulness measure to each of them.

Introduction

The human visual system has the capability of spontaneously detecting many very general classes of patterns in an image, even when there is no high-level or semantic knowledge available to guide the interpretation. Figure 1 gives some examples of the range of this capability, including the detection of colinearity, predominant orientation, bilateral and rotational symmetry, and repetition in an otherwise random field of dots. Computer vision systems currently lack almost all of these early vision capabilities, with the exception of edge detection. Even the edge detection capabilities of current computer programs are far below the level of human performance.

The importance of early vision and spontaneous image organisation has long been recognized, and has gone under names such as image segmentation, figure/ground phenomena, perceptual grouping, and gestalt organisation, all of which emphasize the selection of subsets of image elements which somehow naturally belong together. A grouping is successful to the extent that it brings together elements in the image that have arisen from the same process or belong to the same object in the three-dimensional world being viewed. These groupings can then greatly reduce the combinatorics of the search space when forming higher levels of grouping, matching against world knowledge, making use of texture properties, or searching for correspondences as in stereo vision.

Previous methods for image segmentation have usually been derived from an idealized model of the world (such as the step edge model often used in edge detection or regular texture models used in texture description). The approach taken here is very different and is largely free of prior expectations about the structure of the world. The central concept of this paper is that it is possible to calculate—in a domain-independent way—a statistical measure of the likelihood that some grouping truly reflects an interdependence of its subparts in the world (i.e., that the grouping is not the result of a random alignment of independent elements). This measure can be applied to groupings at all resolutions and positions in the image to determine which ones are the most meaningful and are most likely to lead to further correct interpretations. There is no need to assume a certain level of "noise" in the image, since groupings at all resolutions (allowing all ranges of variations) can be examined, and those which are most meaningful (carry the strongest statistical implications) can be selected as the most useful description of the structure. To give a practical example, it may be important for further interpretation to recognize that the edge of a tree trunk is essentially straight, even though our eye is able to resolve many small perturbations along its length. In this case it would be important to generate at least two different meaningful descriptions for the same curve in the image, corresponding to different resolutions of grouping.

The importance of making precise measurements of the meaningfulness of each grouping is most apparent when attempting to derive strong global information from locally weak information. For example, an edge in a digitized image may be indistinguishable over each small neighborhood along the edge from background sensor noise, as many researchers attempting to derive local edge detectors have discovered. However, if we make many local measurements of meaningfulness at different orientations, and then measure the statistical likelihood that different groupings of local measures would happen to align themselves into a longer smooth edge, we are able to derive much higher measures of meaningfulness for the overall edge than for the local measurements. This implies that a long edge is more detectable than a short edge and that a straight edge

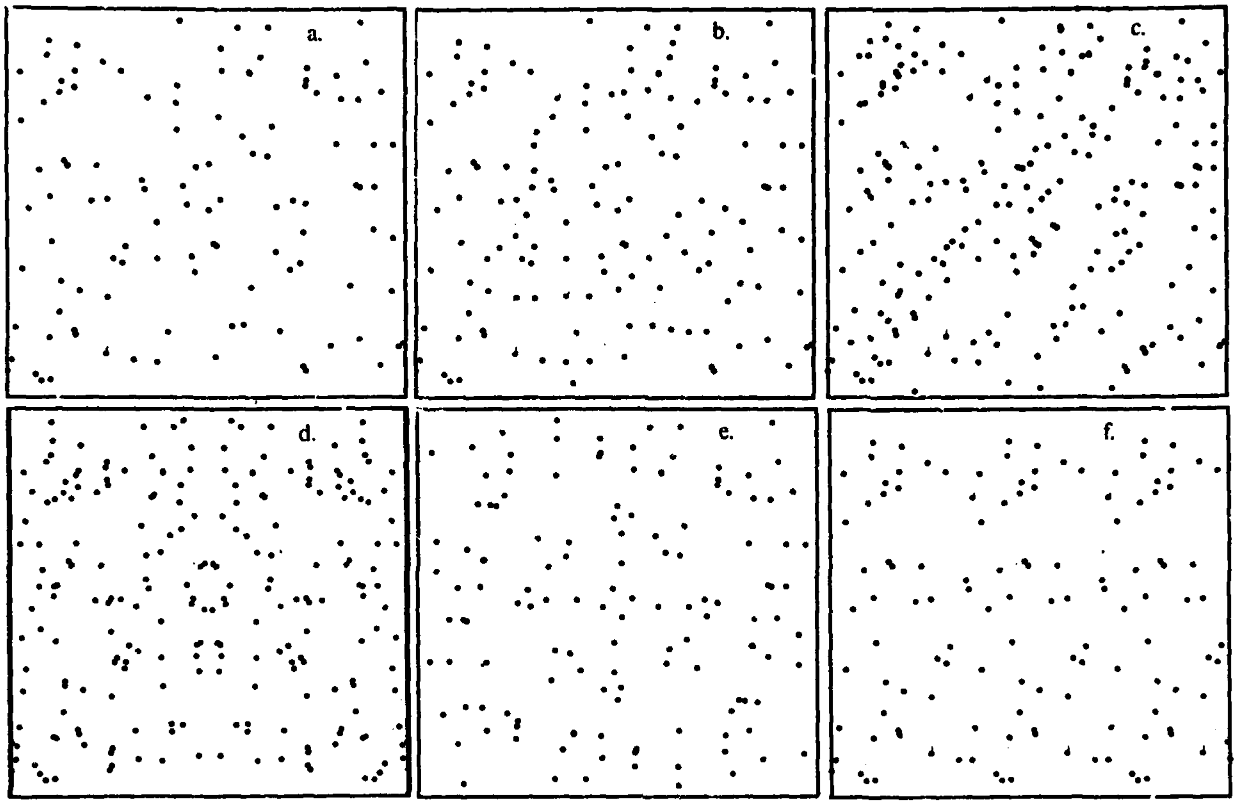


Figure 1: Some examples of the human ability to detect various classes of patterns in an otherwise random field of dots: (a) the basic pattern of 110 dots positioned at random; (b) a string of dots is added which has nearest neighbor distances approximately the same as the background average; (c) the field in (a) is shifted diagonally and overlayed on itself, resulting in a statistically predominant peak in orientation; (d) the field of dots is reflected about a vertical axis, producing bilateral symmetry; (e) the upper right quadrant is rotated into the other quadrants to produce four-fold circular symmetry; (f) the left third of the image is shifted and reproduced twice to form a repetitive pattern.

is more detectable than one with many sharp twists and turns, something apparently also true of human vision. A similar need for global combination of weak local measures applies to all of the examples in Figure 1.

Although the meaningfulness measures applied to each grouping are domain-independent and based on purely geometrical measures of non-randomness, there is still considerable leeway in choosing the groupings to subject to these tests. To test all possible combinations of image elements would be an exponentially expensive process. The human visual system certainly does not recognize all meaningful groupings: as is shown in Figure 2, a pattern such as five equally-spaced dots aligned in a row (which is very unlikely to have arisen by random) will not be spontaneously detected if it is surrounded by enough similar elements. Our approach to defining the sets of groupings to be considered is to use several general principles (such as scale invariance and linear time computational complexity) and to otherwise rely on data regarding human performance.

The Measurement of Meaningfulness

By *meaningfulness* we mean a measure of how likely some grouping is to have arisen from an underlying physical relationship between the constituent features rather than through some accident of viewpoint or location. It is important to have quantitative measures of meaningfulness so that more global combinations of local measures will have precise information to work with in evaluating the significance of each combination. Some commonly used operations, such as thresholding and linear convolutions, destroy a great deal of this information.

We calculate the probability that each selected grouping in the image could have arisen from a random perturbation of the surrounding distribution of similar features. If this is unlikely, then as is commonly done in inferential statistics, we infer that nonchance factors are probably responsible for the grouping. It is common in inferential statistics to specify a threshold at which something is considered to be meaningful (e.g., a 0.05 significance level, or alpha level, corresponding to one chance in 20 that such an unusual result would have arisen from random data). We use a threshold such as this when deciding which results to display during output, but otherwise there is no need for any type of thresholding. The significance values can be combined into higher levels with their own significance based on the fact that two events which are unlikely to have occured at random are even more unlikely to have occured together.

The most common way to measure the likelihood of some value is to look at the shape of the distribution in which it is embedded and measure what proportion of the distribution has values at least as extreme as the one under consideration (for example, we may assume a Gaussian distribution and compare the value to the standard deviation). However, in the types of vision problems we are considering there is no particular knowledge of the shape of the distribution and we do not have enough data to reliably derive the shape from the data for each region. Therefore, we have chosen to use a distribution-free way of calculating likelihoods based on the methods of nonparametric statistics. A nonparametric test of significance makes no assumptions concerning the shape of the parent distribution or population. Its measure of likelihood is based on a *rank test* which measures what proportion of all combinations of rankings have values ranked higher in the surrounding distribution than the values under consideration.

We use discrete statistics above because that is the nature of the vision problem. The image data is not a continuous function but a set of discrete values, and further groupings consist of discrete sets of these initial values. That is one reason why we emphasize the grouping of isolated points in our examples rather than working with gray-scale images that superficially appear continuous--the discrete versions are probably more accurate reflections of what the individual stages of human vision have to work with.

Which Groupings Should be Tested?

As mentioned previously, it would be combinatorially expensive to examine all possible groupings in an image. It would be prohibitive to even form all pairs of features, let alone the larger sets. The human visual system clearly detects only certain classes of meaningful patterns, and experimental work has been carried out to explore this range of performance. Within the computer vision community, Marr [8, 9] in conjunction with Riley [11] and Stevens [13] has discussed the importance of grouping operations and has demonstrated many informal psychophysical experiments testing human performance. Marr emphasized the need for grouping on the basis of length, orientation, size, contrast, and spatial density. However, very little of this work was fully specified or implemented in computer programs. Within the psychology community there have been many investigations of human performance on specific grouping problems. Julesz [6] has carried out many experiments on human performance in distinguishing different textures. Glass [4, 5] has examined the perception of Moiré

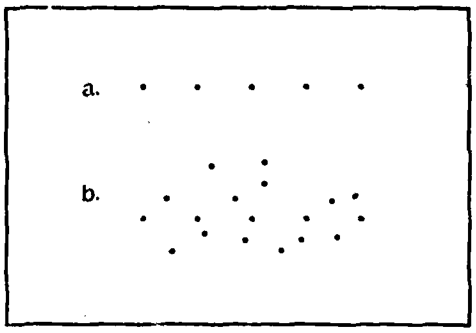


Figure 2: Some patterns, such as the five equally spaced dots in (a) are not spontaneously detected by human vision if they are surrounded by enough similar elements, as in (b), even though the five dots remain highly meaningful in the statistical sense.

patterns of the sort shown in Figure 1c. Many researchers have been intrigued by the human ability to detect bilateral symmetry in otherwise random images (as in Figure 1d), and quantitative experiments on human performance in the face of perturbations in the symmetry have been carried out by Barlow and Reeves [1] and Bruce and Morgan [3]. With the exception of some of Marr's work, all of this research has focussed on performance rather than mechanism.

In addition to data on human performance there are several other constraints on the classes of groupings which should be formed. One is the principle of scale invariance, which means that the same groupings should be formed over a wide range of different sizes in the image. This is just another case of the principle of viewpoint invariance, which also implies the obvious position and rotation invariance. The practical implication of scale invariance is that the same grouping operators must be applied at a range of different sizes in the image (usually chosen to increase by powers of two). Another constraint is that the classes of groupings attempted should be of linear time complexity in terms of the number of items being grouped. Since we are dealing with such large numbers of elements, any attempt at higher order complexity would seem to be too computationally intensive. Therefore, we can only examine groupings between each feature and a fixed number of neighboring features, relying on lower resolution operators to connect features which are more physically distant in the image. When features cannot be grouped at a lower resolution and are too distant to be grouped at a high resolution (as in Figure 2), then the grouping will not be detected.

Only the very lowest levels of grouping operate directly on the image intensity data. Other levels combine the results of previous groupings, looking for meaningful groupings of meaningful values, and in this way build up a layered description of the image. Since there must is some overlap between selected groupings along each dimension in order to minimize the effects of discretization, it is possible to interpolate between neighboring values to precisely locate the best description for each feature.

Linear groupings. One level of image segmentation that has received a great deal of attention is the detection of "edges," usually defined as the detection of extended intensity discontinuities in the original scene. In keeping with our philosophy of looking for statistically meaningful groupings in the image rather than for the image of some idealized feature in the world, we prefer to think of edge detection as the detection of meaningful linear or curvilinear groups of points, where the values of the points have already been detected by some earlier st ge. In this case the earlier stage should probably be an isotropic (circularly symmetric) operator which calculates the ratio of center to surround intensity (the strongest evidence for this stage of processing comes from neurophysiological experiments measuring the output of center-surround neurons in the retina). This isotropic operator would be applied at a range of resolutions over the entire image, and at each resolution we would look at all orientations and positions for linear sets of isotropic values that were significant with respect to the surrounding isotropic values.

When combining meaningfulness values from independent regions of the image (such as when combining the values of isotropic operators that lie in a linear arrangement), the independent likelihood values are multiplied together to produce the meaningfulness measure for the new combination. For example, if there is only one chance in 10 that some feature would arise at random from its surrounding distribution, and there is a likelihood of 5 for some other independent feature aligned with it, then there is only one chance in 50 that a specific combination with such unusual values would have arisen from the distribution. However, if we attempt to make many groupings of some feature with its neighbors, then we must divide the likelihood of the combination by the number of attempted groupings of each feature to compensate for the increased number of groupings being tested. These considerations all derive from basic probability theory when calculating the likelihood that some grouping would have arisen randomly from a background distribution.

In addition to looking for linear groupings at a full range of sizes (resolutions) in the image, we also need to look at a range of elongations (ratios of length to width). We do not know in advance what the length of a linear grouping will be, and attempting to form groupings at all elongations allows us to combine the statistical information over the entire length of an edge before deciding upon its significance. As we increase the elongation it is necessary to increase the number of orientations being examined by the same ratio in order to cover the full space of possibilities. However, longer elongations need to be sampled less frequently in the direction of elongation, so the overall computational requirements remain constant as elongation is increased. This example of detecting linear features is worked out in full in the following section of this paper, and a computer implementation of the algorithm is described.

Some important features of this method for detecting meaningful linear structures are that it does not require edges to be continuous (it works well with dotted lines or

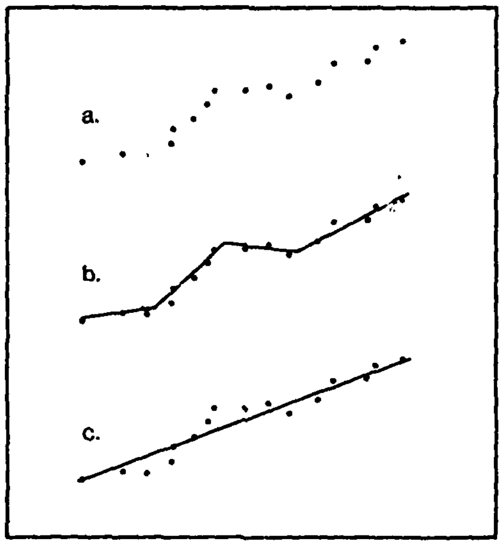


Figure 3: There may be more than one peak in meaningfulness at different resolutions for the same data. The dots shown in (a) could be described as a series of straight line segments at one resolution, as in (b), but also be represented as a single linear grouping at a lower resolution, as in (c).

lines with gaps) and it makes no prior assumptions about the amount of "noise" (variation) in the linear structure. Since it tests for statistical meaningfulness over all possible lengths of an edge, using a much more sensitive test than the typical linear mask, it is possible to accomodate any amount of noise in the linear description so long as the length of the edge is sufficient to make the overall linearity statistically meaningful. After testing all resolutions and elongations, it is possible to select the resolution and elongation with the highest meaningfulness as the most useful description of the grouping. Many groupings may have more than one peak in meaningfulness at different resolutions or elongations, as shown in Figure 3.

Curvature and corners. We have dealt so far with only straight edges. One possible extension would be to apply the same grouping techniques to regions of the image corresponding to all possible arcs of constant curvature. However, the increase in computation this requires is rather large, and human performance does not seem to make full use of the statistical information over the length of a curve of constant curvature (however, careful experiments have not yet been carried out to test this point). It seems more likely that human vision only detects smoothness between locally linear groupings. In other words, a smooth curve contains a sequence of meaningful linear segments—the length of each segment depending on the resolution being used—and neighboring linear segments at slightly different orientations are grouped into a description of curvature at that point. The allowable range of orientations for smooth curvature depends on the elongation of the linear segments,

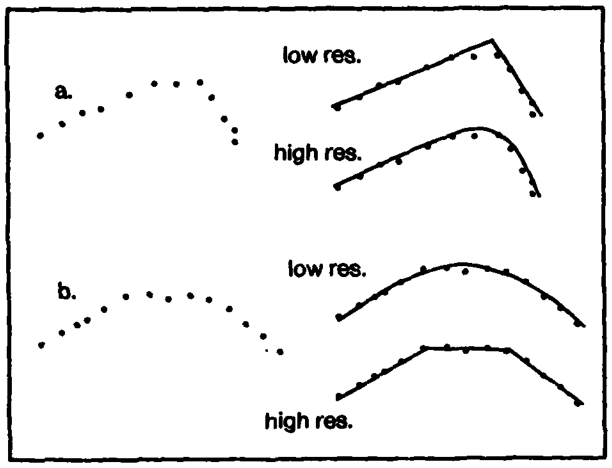


Figure 4: The decision as to whether there is a corner (tangent discontinuity) along a curve can change at different resolutions depending on the length of the support for the curves on each side of the potential corner. In (a) there is a corner when grouping at a low resolution but a smooth curve at a higher resolution. In (b) the roles are reversed.

with longer elongations allowing less change in orientation between segments.

The dual of a smooth-curve continuation from a segment is the detection of a termination. In other words, if there is no smooth continuation from a meaningful linear grouping, then this implies the detection of a meaningful termination of the curve. A corner occurs where two terminations coincide. The same set of points may have a termination at one resolution and may be grouped as a smooth curve at another, as shown in Figure 4.

Marr [8], Schatz [12] and others have shown the importance of detecting terminations and grouping them in further stages. Many of their examples of human texture discrimination are best explained by assuming that "virtual lines" are formed between curve terminations in the same way that we have formed linear groupings of other points in the image. Figure 5a shows an example of an edge formed by a linear sequence of terminations, and Schatz gives many other examples where grouping of terminations is necessary for texture discrimination. Each termination can be treated as a point and fed back into the colinearity detection stage. This is one place where the multiple descriptions of Figure 4 become important, since human vision can detect alignments of low-resolution corners or terminations, even when a curve is smooth at a high resolution.

Orientation and size. Almost all work on edge detection within the computer vision community has dealt with the detection of edges between regions of different intensity. However, as Figure 5 demonstrates, there are many cases in which human vision detects edges between regions with the same average intensity but with properties differing in other ways. One of the strongest effects is produced by changes in orientation, as shown in Figure 5b. The discrimination of the more subtle differences in Figure 5c can be explained similarly by assuming that virtual lines are constructed between the line terminations and the differing orientations of these virtual lines are discriminated. Another significant dimension of variation is size, including both length and width of elongated elements. In Figure 5d each dot in one region is smaller than the other, although their number has been increased to produce equal image intensity in both regions. Figure 5e has the same number and size of dots in both regions, but with different spatial distributions. Discrimination of this example can be explained by the same mechanism as for Figure 5d, since at some lower resolution the dots in the central region will clump into clusters with greater "size" but lower density than the other region. Figure 5f shows the effect of differing lengths, which is much less pronounced.

Size and orientation parameters need to be calculated for regions at all resolutions in the image just as average intensity was calculated, and these results can be fed into the edge grouping process in the same way as the intensity information was. However, there are a number of important issues to be resolved in this process. First of all, how detailed are the characterizations of the distributions of element parameters in each region? It seems that they are not very detailed at all, as is shown in an example by Riley (reproduced in Marr [9]) in which people are unable to distinguish a region with equal numbers of edges at two opposite orientations from a region with randomly distributed orientations. This is another example of a highly meaningful grouping which human vision fails to detect. It appears that a single parameter specifying predominant orientation is all that is required to explain human performance. We intend to carry out a series of similar experiments to precisely determine the characterization that human vision gives to orientation and size statistics. A second important issue is how to determine meaningfulness of these orientation or size measures. For example, if a region only contains one element there is no way to determine whether a particular orientation or size value is random or meaningful. In general, as long as a region contains at least two elements it is possible to calculate some meaningfulness value, and the more elements it contains the higher is the potential meaningfulness.

Symmetry and repetition. Assuming that the above calculations are carried out in a fairly complete way, we conjecture that the detection of symmetry and repetition will require no further mechanisms. Note that the linear grouping of objects as described above will group even a single nearby pair of objects into a linear grouping, although the meaningfulness assigned to a single pair will be low. However, if we look for peaks in the orientation distribution of each region in the image, then a number of low-meaningfulness linear groupings can have a high meaningfulness if they form a large peak in orientation. Repetition of a pattern (including the case where the repeated pattern is reflected about some axis as in bilateral or other symmetries) will result in many parallel matches between similar features at various resolutions. These parallel matches will not be significant unless

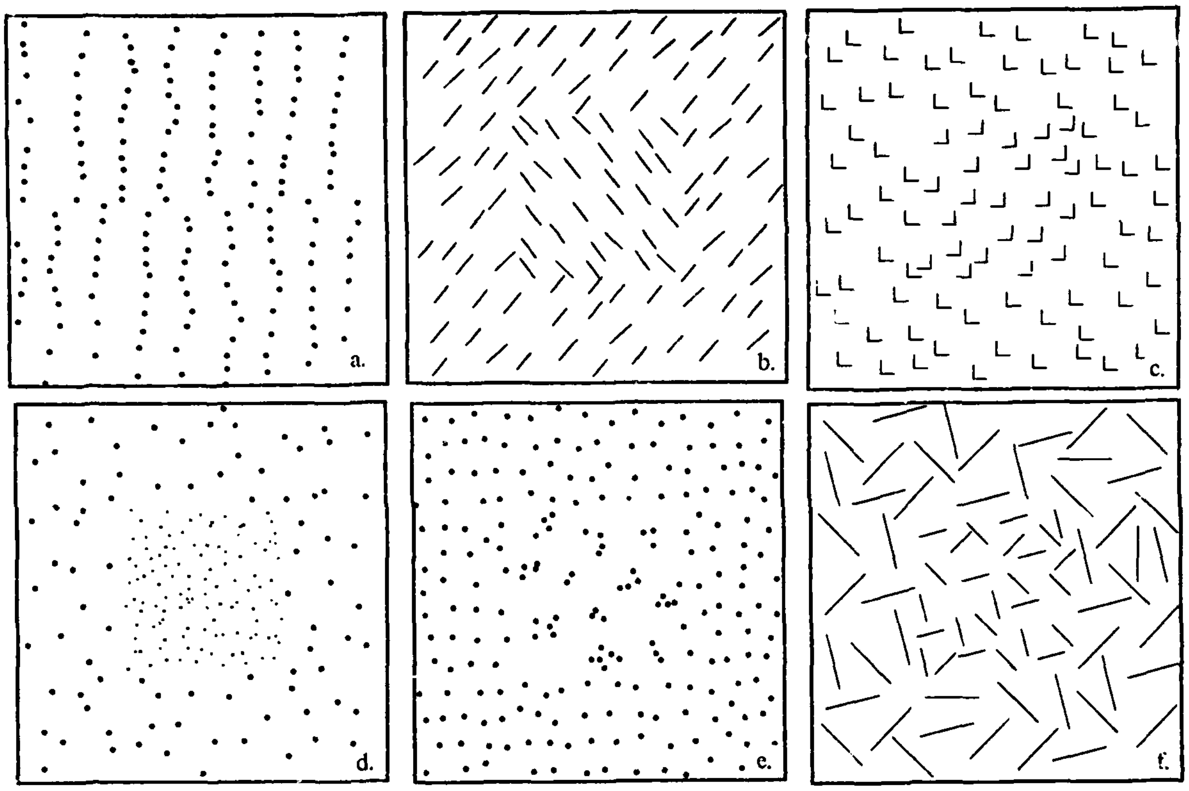


Figure 5: All of these examples have the same average intensity over the differing regions, yet human vision is capable of detecting edges based on changes in other properties. In (a) there is a linear alignment of edge terminations horizontally through the center of the image. Example (b) demonstrates the detection of a change in orientation, and (c) demonstrates a change in orientation of virtual lines. In (d) there is a change in element size, in (e) there is a change in clustering statistics, and in (f) there is a change in line length.

the components are located close to one another at their particular resolution; however, as many psychological experiments have shown (see Julesz [6]) human performance deteriorates very rapidly as the distance to be spanned in making these correspondences increases. Given this conjecture, we intend to carry out other experiments to test its implications for human performance.

The case of symmetry points up the role of high-level knowledge in detecting meaningful groupings. Symmetry is easier to detect when the orientation of the symmetry is parallel to some strong local reference (e.g., the edges of the figure or the page). For operations of the type we have described, the only role that high-level knowledge can play is to bias the meaningfulness results calculated by lower level operations. This can lead to improved performance in such tasks as symmetry detection where the low level results are very weak to begin with. However, this is still very different than the heterarchical approach often adopted in artificial intelligence, which is to calculate only the easiest results bottom-up and to use this knowledge in a top-down fashion to guide computation of the rest. The reason this heterarchical approach fails in many cases for low-level vision is that *all* the low-level results may be weak so that none of them can be used to guide computation of the rest. The only solution in this case is the computationally intensive one we have adopted: form all potential groupings bottom-up and test each one for meaningfulness of the entire combination.

Three-dimensional groupings. It has long been recognized that an important function of early vision is the derivation of the three-dimensional structure of the scene. In previous papers by Binford [2] and Lowe and Binford [7], we have described how various classes of meaningful alignments in a monocular image carry implications for the three-dimensional structure of the scene. For example, if elements of the image are colinear then they are also colinear in three-space, barring an accident in viewpoint. If two edges terminate at a point in the image, or three or more edges converge to a common point, then they must also terminate at a common point in three-space unless the viewpoint is restrictively aligned to produce the coincidence. If one curve terminates at another continuous curve, the terminating curve cannot be closer to the viewer than the continuous curve, or the termination would be unlikely to occur at that location. Curves which are parallel in the image are probably parallel in three-space. There are other similar inferences for interpreting cast shadows or the boundaries of a region. In all these cases, the inferences are based on measures of meaningfulness, where a meaningful grouping in the image leads to statistical inferences for the three-d[illegible]sional structure. By combining the meaningfulness of ima[illegible] groupings as described above with the assumptions of general camera position and general light-source position, it is possible to precisely quantify the strength of each inference.

One much studied mechanism for deriving three dimensional information is stereopsis, which depends on a general-purpose mechanism for matching between two images. The groupings we have described can serve as preliminary descriptions for forming matches between images, and the resulting matches can be treated as new groupings with meaningfulness measures of their own. For example, in performing stereo interpretation of random-dot stereograms, at each resolution about one element in 10 should form a grouping with 0.1 significance, one element in 100 should form a grouping with 0.01 significance, etc., and if two images contain significant features from the same class within small corresponding fusional areas, it would be possible to calculate the meaningfulness of this correspondence. Mayhew and Frisby [10] describe a stereo interpretation system that detects edges by grouping isotropic point measures in linear three-space groupings, similar to the techniques outlined here but without an explicit measure of meaningfulness. Stereo processing presents yet another opportunity to derive highly meaningful larger groupings from weak local groupings, in addition to its role in providing explicit depth information.

Implementation of the Linear Meaningfulness Algorithm

In order to illustrate these methods in more detail, we have written a program for detecting meaningful linear groupings of points in an image at all resolutions, orientations, and elongations.

The program takes as input a set of dots like those in Figure 9a (although they need not all be of the same size). The first stage of the program accumulates the density of points falling into square regions at all resolutions from 1/256 the width of the image to 1/8 the width of the image, each resolution twice as course the previous one (a total of 6 resolutions). Each region overlaps with its neighbors by 50% in the vertical and horizontal directions, so that each point falls into four of these regions at each resolution. We then compute the center-surround values at each resolution, which is done by subtracting the average intensity of the surrounding 8 square regions from the intensity of each central region. This first stage of processing in our implementation is very crude—there should at least be smooth transitions between neighboring regions rather than abrupt boundaries. However, the novel part of the algorithm is not in this stage but in the way these initial isotropic values are combined to produce meaningful linear groupings.

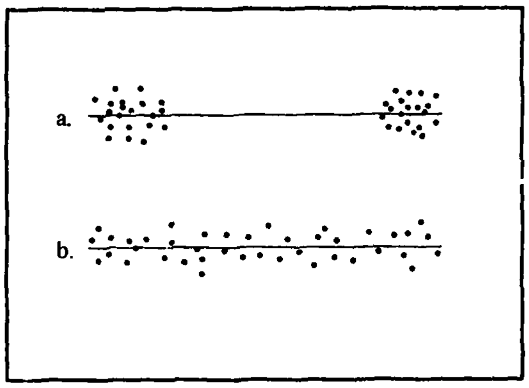


Figure 6: These two sets of points have the same standard deviations from best-fit lines of the same length, yet (b) is much more meaningful as a linear feature than (a).

One commonly-used method for measuring the degree of linearity among data points is to measure the standard deviation of the data points from the line with the best least-squares fit, with a lower standard deviation indicating a better fit. However, as Figure 6 demonstrates, two groups of points with the same standard deviation to a line of the same length can exhibit very different degrees of meaningful linearity. That is the importance of the two-stage process we have used here, where isotropic meaningfulness values are calculated first and their effects separated from the linear meaningfulness values.

As discussed in the previous section, we examine sets of isotropic values in linear arrangements in the image and calculate the probability that each set of such meaningful values would arise at random. This is done by first comparing each isotropic value to a surrounding set of similar values, and calculating a likelihood for it based upon its rank in the surround. Then, given that the isotropic values are computed over independent regions of the image, we multiply their likelihood values together to compute the likelihood that values with a meaningfulness at least that high would happen to occur together. Finally, we divide this value by the number of groupings of that class attempted from each point, since the more groupings which are attempted the more coincidences we expect to find in a random distribution.

We start by computing linear meaningfulness of just pairs of isotropic values and then combine these in stages into longer elongations. We compute linear meaningfulness for pairs at eight orientations as shown in Figure 7, following our rule that adjacent orientations should overlap by 50% to minimize the effects of discretization. We compare each value in these pairs to those of eight surrounding values in sidebars alongside the pair as shown in Figure 8. Meaningfulness is calculated by taking the total number of surrounding values plus one and dividing by the number which are ranked higher than the center value plus one. Therefore, if one of the center values is higher than all 8 of

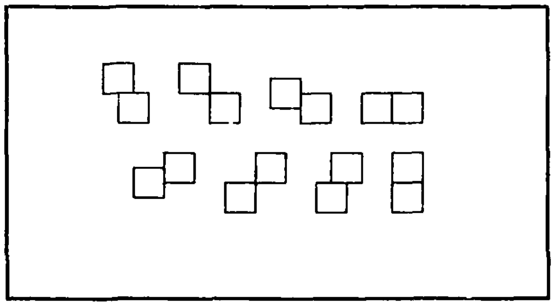


Figure 7: Meaningfulness values for pairs of isotropic values are computed at eight orientations, making use of the independent overlaps in two directions.

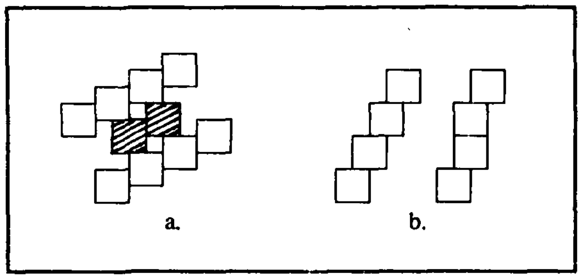


Figure 8: The meaningfulness of each pair of isotropic values is computed with respect to eight surrounding values as shown in (a). In (b), a pair is combined with two colinear pairs of the same orientation but slightly different positions.

the surrounding values, it is assigned a meaningfulness of $(8+1)/(0+1) = 9$, whereas if it is only higher than seven of them it is assigned a value of $(8+1)/(1+1) = 4.5$. Note that one of these linear operators can be assigned a certain degree of meaningfulness just from a single meaningful isotropic value—this is necessary so that combinations into longer elongations will have some measure by which to compute the meaningfulness of the combination even if there is incomplete evidence for linearity in the shorter operator.

Each of these linear operators is combined with its neighboring colinear operators to produce operators with longer elongations. Each stage of combination produces elongations which are twice as long and therefore need to be tested at twice as many orientations to cover the space of possibilities with the same degree of overlap. Therefore, each linear operator is combined with each of two operators with the same orientation but slightly different positions perpendicular to the direction of linearity—to approximate linear operators of different orientations—as is shown in Figure 8b.

The results of carrying out these computations on an image of dots is shown in Figure 10. Figure 9a is the original input to the program, which contains a linear feature immediately apparent to human vision but which nonetheless is based on weak local evidence. Figures 10a though 10c show the results of computing linear meaningfulness at three different resolutions, each one twice as course as the previous ones. Each line shows position and length of a linear operator and the circles at the end of each line are proportional to the log of the likelihood computed for that operator. The large amount of output in these figures includes many linear groupings which are only of marginal meaningfulness, although data of this sort would be necessary in many situations for grouping on the basis of orien-

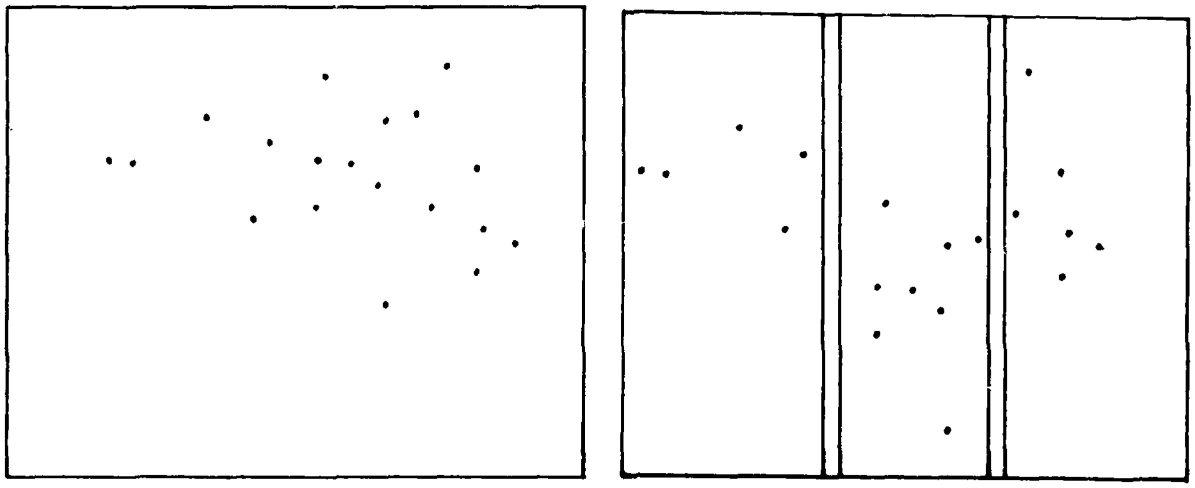


Figure 9: The dots shown in (a) are the input for the computation shown in Figure 10. Although there is a prominent linear feature in (a), this feature is not easily detectable on the basis of local evidence if we separate the image into thirds and displace the center third downwards, as shown in (b).

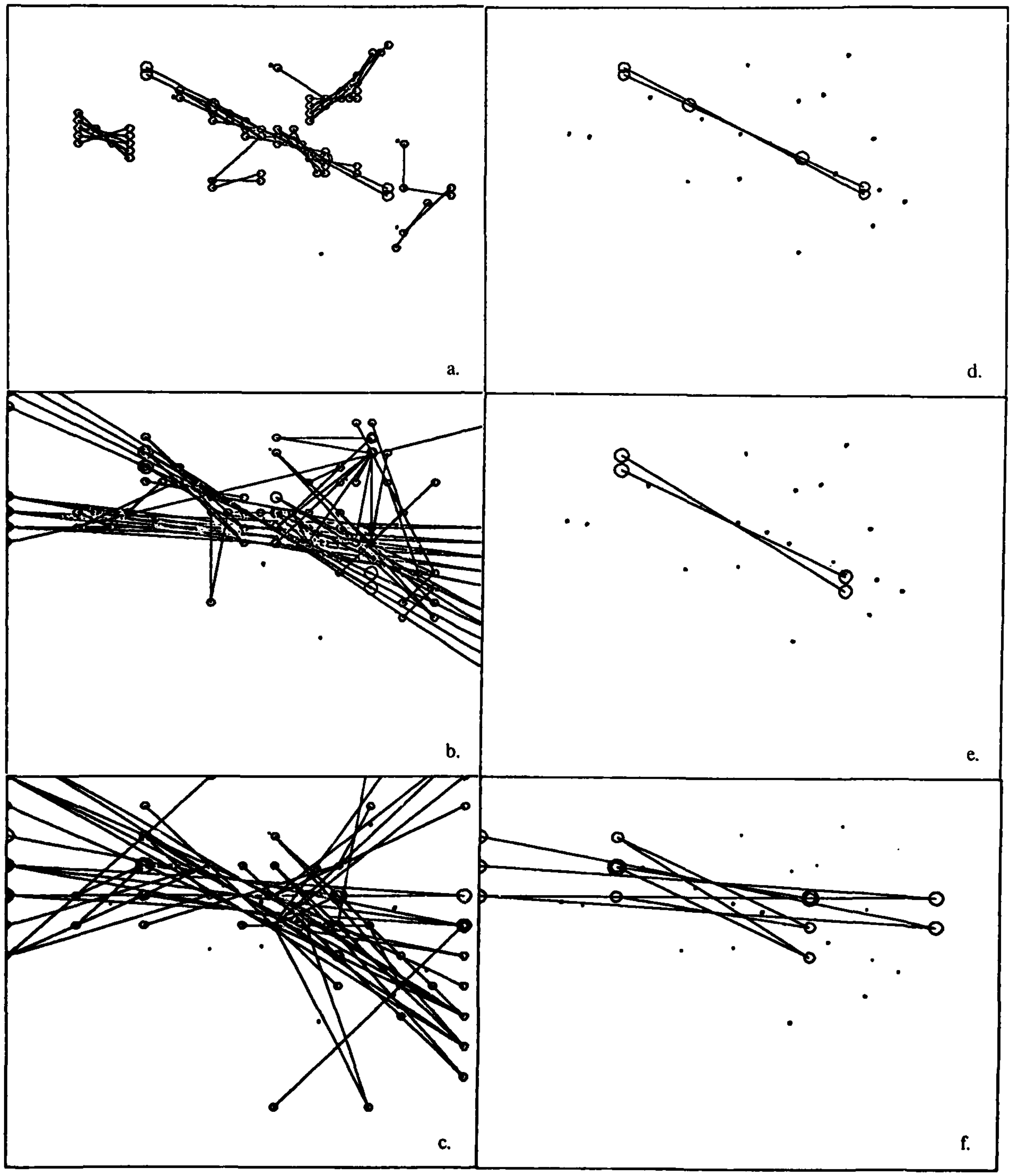


Figure 10: Figures (a) through (c) show the results of computing meaningful linearities at three different resolutions (of increasing powers of 2) on the data of Figure 9. The lines represent linear groupings at various elongations, and the circles at the end of each line are proportional to the log of the likelihood value. If we reduce the significance threshold to the 0.01 level, we are left with the results displayed in (d) through (f).

tation or other higher level grouping. However, when we reduce the meaningfulness threshold for display to the 0.01 significance level, we get the results shown in Figures 10d through 10f, which give only linear groupings of elements corresponding to the single prominent diagonal line.

This initial demonstration is quite crude in many respects, and there are many issues remaining to be resolved. Nevertheless, there are some ways in which this example has impressive performance. As shown in Figure 9b, if we examine small regions of the input data there is insufficient information to reliably detect a linear arrangement of dots. Therefore, the program has in some sense combined the statistical information from along the full length of the feature before arriving at its conclusion of meaningfulness. This is a simple example of the detection of globally significant features from locally weak information that was discussed earlier. Also, unlike most other edge detection programs, it has made no assumptions about the amount of "noise" (deviation of the dots from a straight line), and would have detected the grouping over a very wide range of sizes in the image. Unlike linear convolutions, it is relatively insensitive to a few large dots placed near the linear grouping in the image, since it uses a non-parametric rank test rather than making assumptions about the surrounding distribution. This approach is also very different than the simple detection of zero crossings, since zero crossings are a local measure that may not be supported by any meaningful information (for example, zero crossings may wander randomly under the influence of sensor noise in regions of the image which do not have some sufficiently strong changes in intensity).

Computation Time Considerations

The methods discussed in this paper—testing groupings for meaningfulness at all possible positions, resolutions, and parameter values—are more computationally expensive than those used in most current computer vision programs. However, the difference is only a moderate linear increase over other methods and is not as large as might be feared at first. For example, examining the image at 6 resolutions can be less than a factor of 2 more expensive than examining it at the finest resolution, since halving the resolution means that only one quarter as many groupings need to be examined over the area of the image. When using a digital computer there are techniques that allow us to not even consider groupings that do not have any potentially meaningful constituents. The program described in the previous section used hash coding to access each grouping (each dot was hashed into all groupings which contained it), so that groupings for positions in the image which did not contain any dots were not even considered.

However, regardless of these implementation considerations, it seems likely that there is no alternative to examining these large numbers of groupings if machine vision is to rival human visual capabilities. As emphasized earlier, there may be no information available at an early stage to indicate which of a large number of possible groupings will turn out to be meaningful. From what we know of the human visual system, it seems that our brains have opted for a brute-force, parallel approach to carrying out these computations. A surprisingly large proportion of the human brain seems to be devoted to simply computing local results over all positions in visual images.

Summary

Our derivation of the function of early vision does not start from a specific model of what we expect to find in the world. Given that the world is very general and variable, any prior knowledge about its structure at this level would probably be rather weak. Instead we see the task of early vision to be the formation of meaningful groupings in the image, where meaningfulness can be tested in a domain-independent self-verifying way. It is possible to approach this task as a signal detection problem which makes maximum possible use of the statistical information in the image to distinguish significant relationships from the background of accidentals.

Therefore, meaningfulness does not depend on prior world knowledge. For example, we emphasize the detection of linearity as one useful basis not because it is a common structure in the world so much as because it is a particularly simple basis, where simplicity is required to make maximum use of the available information. Therefore, linearity is a useful way to segment and describe random fields (for the purpose, say, of comparing them to similar random fields) even though the fields were generated without respect to any linearities. Although we cannot claim to have fully done so in this paper, it seems possible to base this method on a sound mathematical derivation.

One result of the domain-independence assumption is that control issues become less important. We compute all possibilities in a bottom-up exhaustive manner, and have given reasons why there may be no computationally less-expensive way to achieve these results. For example, there is probably nothing to gain from the often mentioned technique of detecting the strong parts of an edge and "extending" these segments to look for weaker segments, since we also want to be able to detect the edge when all parts are locally weak. Further progress in low-level perception is more likely to come from attempts to precisely specify what we want to measure than from improvements in control of the computation.

Of course, the above is not meant to imply that all aspects of early vision can be derived from a few abstract principles. Fortunately, we are working in an area in which it is comparatively easy to perform experiments to test the capabilities and parameters of the human visual system. The measurement of meaningfulness we have described is a theory that can be subjected to empirical tests and can be used to guide experimentation. One encouraging result is that a few fairly simple computations seem to cover what initially appear to be a very diverse set of capabilities. There are numerous avenues along which to pursue further research.

Acknowledgements

We would like to thank David Marimont, Brian Wandell, and our other colleagues for their help. This work was supported under ARPA contract N00039-82-C-0250. The first author was also supported by a postgraduate scholarship from the National Research Council of Canada.

References

[1] Barlow, H.B. and R.C. Reeves, "The versatility and absolute efficiency of detecting mirror symmetry in random dot displays," *Vision Research,* **19** (1979), 783-793.

[2] Binford, Thomas O., "Inferring surfaces from images," *Artificial Intelligence,* **17** (1981), 205-244.

[3] Bruce, Vicky G. and Michael J. Morgan, "Violations of symmetry and repetition in visual patterns," *Perception,* **4** (1975), 239-249.

[4] Glass, Leon, "Perception of random dot interference patterns," *Nature,* **246** (1973), 360-362.

[5] Glass, Leon and Eugene Switkes, "Pattern recognition in humans: Correlations which cannot be perceived," *Perception,* **5** (1976), 67-72.

[6] Julesz, B., "Experiments in the visual perception of texture," *Scientific American,* April, 1975.

[7] Lowe, David G. and Thomas O. Binford, "The interpretation of three-dimensional structure from image curves," *Proceedings IJCAI-7,* (Vancouver, Canada, August 1979), 613-618.

[8] Marr, David, "Early processing of visual information," *Philosophical Transactions of the Royal Society of London, Series B,* **275** (1976), 483-524.

[9] Marr, David, *Vision,* (San Francisco: W.H. Freeman, 1982).

[10] Mayhew, John E.W., and John P. Frisby, "The computation of binocular edges," *Perception,* **9** (1980), 69-86.

[11] Riley, M., "The representation of image texture," Master's Thesis, MIT, September 1981.

[12] Schatz, Bruce R., "The computation of immediate texture discrimination," MIT AI Memo 426, August, 1977.

[13] Stevens, K.A., "Computation of locally parallel structure," *Biological Cybernetics,* **29** (1978), 19-28.

AD-P000 124

Incremental Acquisition of a Three-dimensional Scene Model from Images

Martin Herman
Takeo Kanade
Shigeru Kuroe

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

We describe the current state of the **3D Mosaic** *project, whose goal is to incrementally acquire a 3D model of a complex urban scene from images. The notion of incremental acquisition arises from the observations that (1) single images contain only partial information about a scene, (2) complex images are difficult to fully interpret, and (3) different features of a given scene tend to be easier to extract in different images because of differences in viewpoint and lighting conditions. In our approach, multiple images of the scene are sequentially analyzed so as to incrementally construct the model. Each new image provides information which refines the model. We describe some experiments toward this end. Our method of extracting 3D shape information from the images is stereo analysis. Because we are dealing with urban scenes, a junction-based matching technique proves very useful. This technique produces rather sparse wire-frame descriptions of the scene. A reasoning system that relies on task-specific knowledge generates an approximate model of the scene from the stereo output. Gray scale information is also acquired for the faces in the model. Finally, we describe an experiment in combining two views of the scene to obtain a refined model.*

1. Introduction

The goal of the **3D Mosaic** project is to automatically acquire a detailed 3D description (or model) of a complex urban scene from images. We are currently working with aerial photographs of Washington, D.C. Fig. 1 shows a stereo pair of images from our database.

Our approach to this problem is based on the notion of incremental acquisition of the scene model. A single image or view of a complex scene is generally not adequate for deriving a complete, accurate description of the scene. Some reasons for this are:

1. Many surfaces in the scene are occluded in any particular view.

2. Because of the complexity of an image, it would be difficult to interpret all the detailed parts.

3. Some characteristics of visible surfaces may not be as apparent in one image as in a different image. For example, it may be difficult to analyze a highly oblique surface because of lack of resolution in the image, or it may be difficult to analyze surfaces with shadows cast across them.

4. Errors in analyzing and interpreting the image may create errors and inconsistencies in the scene description.

Our method involves using multiple views of the scene in a sequential manner. A partial description is derived from each view. As each successive view is analyzed, the model of the scene is incrementally updated with information derived from the view. The model is initially an approximation of the scene, and becomes more and more refined as new views are processed. At any point along its development, the model should be usable for the following types of tasks:

1. When information is derived from a new view, it must be matched to the model so that updating can occur. The model should, therefore, contain information that facilitates this matching.

2. The model should permit higher-level components to determine which parts of the scene should be analyzed in more detail, and whether a different view is required for further analysis of these parts.

3. The model should be usable in its task domain, e.g. for photointerpretation or display generation.

In our approach, 3D features that are relatively inexpensive to obtain, such as certain corners of buildings, are extracted from the images. A model of the scene is then hypothesized from these features by utilizing task-specific knowledge (e.g. block-shaped objects in an urban scene). Updating and refinement of the model is facilitated by remembering which parts of the model have been hypothesized and which parts have been directly derived from the images.

There are several applications we have in mind for the types of models that are acquired. The first involves model-based photointerpretation. A scene model can provide significant help in interpreting images of the scene taken from arbitrary viewpoints [4, 15]. Furthermore, the analysis results can be used in the incremental acquisition loop to update and refine the model. Anothe area of application deals with generating flight plans (simulating the appearance of the scene along potential flight paths) or familiarizing personnel with a given area. Our methods provide the ability to acquire a model of a scene from only a few views and then generate arbitrary views from the model. Finally, our incremental 3D Mosaic approach should be applicable to robot navigation and manipulation tasks. The ability to incrementally acquire approximate descriptions of complex environments could prove useful for these tasks, since these descriptions may then be used to make decisions dealing with path planning or determining which parts of the environment to analyze in more detail.

In the rest of this paper, we first discuss how to extract a 3D scene description from a single view. The stereo pair of images shown in Fig.1 constitutes the single view to be considered. Afterward, we discuss combining information from multiple views.

2. Stereo Analysis

Our current method of extracting 3D shape information from the images is via stereo analysis. In the future, we may add other methods, such as shadow analysis [16].

Our approach to the stereo matching problem is to match junctions and lines found in the images. There are several reasons for this:

1. Our goal is to recover the 3D structure in the scene. We approach this problem by first extracting 3D information dealing with vertices and edges in the scene. In an urban scene, vertices often correspond to corners of buildings. Therefore, by recovering scene vertices and edges that emanate from them, we obtain portions of boundaries of the buildings. These boundaries then allow us to construct 3D approximations of the buildings. (See [10] for a different approach developed for the same task domain.)

2. Our stereo images are fairly wide angle and the scene consists of tall buildings. As a result, there are large discontinuities in disparity and the appearance of many objects differ significantly in the two images. This has caused problems for most previous stereo matching techniques since there are large portions of the scene that are visible in one image but not in the other. In our approach, we are not interested in matching scene faces that are occluded in one of the image pairs. Rather, our goal is to match face boundaries that are visible in both images. We do this by explicitly taking into account the way junctions change from one image to the other. We find a junction in one image, use task-specific constraints to predict its appearance in the other image, and search for the corresponding junction by making use of the predicted appearance. Currently, our method of predicting junction appearances is based on the following task-specific knowledge:

   a. In aerial photography, image planes tend to be almost parallel to the ground plane.

   b. In urban scenes, roofs of buildings tend to be almost parallel to the ground plane, while walls tend to be perpendicular to this plane.

   c. Therefore, features lying on a roof or on the ground will maintain the same shape in both images. Edges in the scene that are perpendicular to the ground plane will appear in each image to be directed toward the origin, defined by the intersection of the camera axis with the image plane [11].

   If an L junction is found in one image, it is initially assumed to arise from a corner of a roof, and thus its appearance in the other image can be predicted. If an ARROW or FORK junction is found, the line directed toward the origin is initially assumed to arise from a scene edge which is perpendicular to the ground, while the other two lines of the junction are initially assumed to arise from scene edges lying on a roof or on the ground. Again, its appearance can be predicted.

3. Many stereo systems have trouble with wide angle stereo images because they rely heavily on local similarities in the two images [2, 3, 9, 12, 13]. In our approach, however, because the junction is intended to represent a structural component in the scene, we also rely on more global, structural similarities in the two images to perform the matching.

4. For a scene with many occlusion boundaries, an approach based on feature matching results in much more accurate 3D positions for these boundaries than an approach based on gray scale area matching.

2.1. Steps in Stereo Analysis

<u>Extracting lines.</u> The first step in the stereo analysis is to extract linear features. A 3x3 Sobel operator is used to extract edge points, as shown in Fig. 2. Then the edges are thinned using a modified Nevatia and Babu algorithm [14], as shown in Fig. 3. The resulting

edge points are linked and straight lines are fitted to them. The method used to fit straight lines to a set of linked points is based on iterative end-point fitting [7]. However, since this method determines a line using only two end points, the line equation for the set of points is recalculated using least squares. Finally, short lines are discarded. The resulting line images are shown in Fig. 4.

Extracting junctions. The next step is to extract junctions from the line images. A junction is a group of lines that meet at a point, and often arises from a vertex in the scene. We consider the following four junction types: L, ARROW, FORK, and T. To find junctions, a 5x5 window around each end point of each line is searched for ends of other lines. Lines in the window that are close and nearly parallel are combined into a single line. Then, if the window contains the ends of three lines, the lines are classified as an ARROW, FORK, or T junction depending on the angles between the lines. The position of the junction point is the middle of the three end points. If a window contains the ends of two lines, the lines are classified as an L junction. The intersection of the two lines determines the position of the junction point. If a window contains more than three lines, each set of two lines is assumed to form a distinct L junction. Junctions that have been found in this manner are labeled in Fig. 4.

Find potential junction matches. The next step in the stereo analysis is to match the junctions found in one image with those in the other. Let us consider how L junctions are matched. As explained previously, each L junction in one image is initially assumed to lie on a plane which is almost parallel to the camera image planes. The shape and orientation of its corresponding junction in the other image, therefore, can be predicted. Each L junction in the first image may be matched with several junctions in the second image that lie along the corresponding epipolar line and that have, within tolerance, the predicted shape and orientation. An interesting point is that we do not try to match only with junctions in the second image that have been previously found. Rather, the shape and orientation of the corresponding junction in the second image is predicted for every point lying on the epipolar line (on the appropriate side of the infinity point), and at each of these points, a search is made within a pre-specified window for lines that might correspond to the predicted junction. The requirements, however, for two lines to be a junction is more relaxed than the requirements during initial junction search. We therefore improve feature detection in each image by using the features found in one image to predict features in the other image. (Matching is performed in two directions, from the first image to the second, and vice versa.)

To match ARROW, FORK, and T junctions, each pair of lines forming the junction is treated as if it were an L junction and matched in the manner described above.

Search for unique junction matches. Next, a beam search [15] is used to arrive at a unique combination of junction matches. There are two factors involved in computing costs for the various combinations of matches:

1. Local cost between two potentially matching junctions is computed by the similarity of the image intensities inside the junctions. The assumption here is that the two junctions will have similar intensities if they arise from the same face corner.

2. Global cost is based on the consideration that if there are two vertices in the scene with the same heights, the positional relationship between their corresponding junctions in one image will be the same as in the other image. This is due to the image planes being parallel to the ground plane. We make the assumption that junctions which are close to one another will often correspond to vertices lying on top of the same building, thus having approximately the same height. Global cost between two potentially matching junctions is therefore computed by the similarity of the configuration of the neighborhoods around the junctions.

The matching procedure is applied from the first image to the second and vice versa. The results are then merged. Fig. 5 shows junctions and lines in one image that have matches in the other image.

Search for third legs of junctions. The next step is to find lines in the images that might be the third leg of matched junctions and that might represent scene edges perpendicular to the ground plane. The method used is to find lines near the junctions in both images that are directed toward the origin.

Generate 3D wire frames. Finally, 3D coordinates are derived using triangulation. Fig. 6 shows a perspective view of the 3D vertices and edges that result. We call this a wire-frame description of the scene.

3. Representing and Modifying the 3D Scene Model

Our requirement that the scene model is to be incrementally acquired leads to several issues: (1) representing partial constraints on 3D structure, (2) incremental accumulation of these partial constraints, and (3) handling discrepancies in information acquired at different times.

Our approach involves representing the model in a modular manner. Constraints on 3D structure are represented in the form of a graph, called the *structure graph*. The nodes and links represent primitive topological and geometric constraints. The structure graph is incrementally constructed through the addition of these constraints. As constraints are accumulated in the graph, their effects are propagated to other parts of the graph so as to obtain globally consistent interpretations.

3.1. Representation of Model

The current structure-graph representation models surfaces in the scene as polyhedra. The components of a polyhedral surface are the face, edge, and vertex. We distinguish the topology of the polyhedral components from their geometry [1, 8]. The geometry involves the physical dimensions and location in 3-space of each component, while the topology involves connections between the components.

In the structure graph, nodes represent either primitive topological elements -- faces, edges, vertices, objects, and edge-groups (which are rings of edges on faces) -- or primitive geometric elements -- planes, lines, and points. Vertex, face, and edge nodes are tagged as either *confirmed* or *unconfirmed*. Confirmed means that the element represented by the node has been derived directly from the images. Unconfirmed means that the element has only been hypothesized.

The primitive geometric elements serve to constrain the 3-space locations of faces, edges, and vertices. Plane and line nodes contain plane and line equations, respectively. Point nodes contain coordinate values. The graph contains two types of links: the *part-of* link, representing the part/whole relation between two topological nodes, and the *geometric constraint* link, representing the constraint relation between a geometric and topological node.

3.2. Modifications to Model

Modifications to the model will occur as part of the process of incremental construction. Deletions and changes are made when new information is found to conflict with information currently contained in the model. This will happen most often with portions of the model that have been hypothesized. Additions to the model are made to incorporate the new information as part of the model.

Modifications to the structure graph are made by adding or deleting nodes and links, or changing the equations of line and plane nodes, or the coordinates of point nodes. All effects of modifications are propagated to other parts of the graph.

As an example, consider adding or deleting a geometric constraint link between a geometric and topological node. Any of the three geometric nodes -- points, lines, and planes -- may constrain any of the three topological nodes -- vertices, edges, and faces. Fig. 7 shows how a constraint on one node may propagate to others. The arrows in the figure indicate the direction of propagation. For example, if a point constrains a vertex, it must also constrain all edges and faces containing that vertex. Similarly, a point that constrains an edge also constrains all faces containing that edge.

When a geometric constraint link is deleted, the rest of the structure graph must be made consistent with this change. Our approach to this problem is based on the TMS system [6], using the notion that when an assertion is deleted, all assertions implying it and all assertions implied by it should also be deleted, unless they have other support. Assertions that imply a given assertion are obtained by following backwards along the arrows in Fig. 7. Assertions implied by a given assertion involve following forward along the arrows.

Consider the example in Fig. 8a, which depicts three topological nodes (vertex *v*, edge *e*, face *f*) constrained by one geometric node (point *p*). Suppose now that link 4 is deleted (Fig. 8b), i.e.,the assertion "*p* constrains *e*" is deleted. To find the assertion that might imply this one, locate the box in Fig. 7 that represents a point constraining an edge, follow backwards along the arrow, and the result is the box that represents the point constraining any vertex of the edge. In Fig. 8b, this represents the assertion "*p* constrains *v*, and *v* is part of *e*". This assertion must therefore be made false. To do so, we may delete either link 1, link 3, or both from Fig. 8b. We have arbitrarily decided that part-of links should dominate constraint links, and thus link 3 is deleted. This seems to work well for our examples.

We now must determine the assertions implied by the one initially deleted. We follow forward along the arrow from the box in Fig. 7 that represents a point constraining an edge, and the result is the box that represents the point constraining all faces containing the edge. In Fig. 8b, this represents the assertion "*p* constrains *f*", which is link 5. This link should therefore be deleted because it has no other support. The resulting structure graph is depicted in Fig. 8c.

4. Generating the 3D Scene Model

We now present an example showing how the scene model is generated from the output of the stereo analysis component. We start with the 3D wire-frame description shown in Fig. 9. The final model derived is a surface-based description.

Combine edges. First, if there are two wire-frame edges that are *nearly parallel and very close to each other, they are merged into a* single edge. This occurs only once in Fig. 9, for the two edges labeled E1 and E2.

Generate web faces. Next, each vertex is assumed to correspond to a corner of an object. Therefore each adjacent pair of legs ordered around the vertex corresponds to the corner of a planar face. Thus far in our experiments, we have dealt only with trihedral vertices. In this case, every pair of legs of each vertex corresponds to the corner of a separate face. A partial face, called a *web face*, is generated for each such pair.

Merge partial faces. After all web faces have been created, those that represent the corners of a single face are merged. Two partial faces that contact each other (e.g. F1 and F2 in Fig. 9) should be merged if (1) they share exactly one edge, (2) the edge serves as a boundary of both faces, but does not partition them, and (3) the planes of the faces are nearly parallel and very close to each other.

Two partial faces that do not contact each other (e.g. F3 and F4 in Fig. 9) should be merged if (1) each face has a single chain of edges that is not closed, (2) each of the two end points of the edge chain of one face must be uniquely matched with those of the other face, where unique matching is determined by the distance between the two points being less than a threshold, and (3) the planes of the faces are nearly parallel and very close. When merging the two non-contacting faces, the two edges on which each matching pair of end points lie are extended in space and intersected. The intersection points form two new vertices on the resulting face.

Complete the shapes of faces. After all mergers have been performed, many faces may still be incomplete, i.e., they do not have a closed boundary. In these cases, task-specific knowledge is used to hypothesize the shape of each face, and it is completed by generating the appropriate edges and vertices. The rules used here are:

1. If the partial face consists of a single corner, i.e., it contains only two connected edges, the shape is completed as a parallelogram.

2. If the partial face contains three or more edges connected as a single chain, the shape is completed by connecting the two end points of the chain with a new edge.

Find holes in the faces. After all faces have been completed, one face is assumed to represent a hole in another face if (1) the planes of the faces are nearly parallel and close to each other, and (2) the boundary of the first face, when projected onto the plane of the second face, falls inside the boundary of that face. When these conditions are met, the bounding edges of the first face are converted into an inner ring of edges of the second face.

Generate vertical faces for incomplete objects. At this point, many objects will be only partially complete because they are not closed. Task-specific knowledge may be used to add more faces to the object. Because our 3D information is obtained from aerial images of an urban scene, many faces that lie high enough above the ground plane represent roofs of buildings. For each such face, vertical walls are dropped toward the ground plane from each edge of the face, unless the edge is also part of another face. The equation of the ground plane is currently interactively obtained. A vertical wall is dropped either down to the ground plane, or to the first face it intersects on the way down.

Our procedure for dropping vertical faces from a face F is as follows. First, an edge is dropped from each vertex of F either to the ground plane or to the first face it intersects. Next, web faces are created for each new edge pair at each vertex. Newly created faces are then merged and completed in the ways described above. Fig. 10 shows several perspective views of the resulting scene model.

4.1. Comparison with Depth Map

There are several interesting points about the generated model. First, notice that it is a higher level description than a depth map. The product of most stereo analysis systems is a depth map [2, 13] which, like an image, is an array of numbers that requires description. Our approach, on the other hand, has been to extract a sparse amount of 3D information using stereo analysis (as shown in Fig. 9) and to use task-specific knowledge to go directly to a higher level 3D description. This description is much more compact than one based on surface points, and allows properties such as topology, shape, absolute size, and absolute position of scene objects to be easily available. It should therefore be easier to update and refine the model from information obtained from subsequent views. Furthermore, the model should be more useful for matching with 2D image information, with 3D information extracted from images, and with other models.

4.2. Mapping Gray Scale onto Faces

In order to render more realistic displays, gray scale is added to them [5]. This is accomplished by associating with each face in the model a normalized intensity image patch of the face. Although these patches are currently derived from a single image of the scene, we plan to generate them from multiple images. Geometric normalization, which eliminates the effects of perspective projection, is performed on the patches. We also hope to perform photometric normalization to eliminate the effects of varying illumination conditions. Fig. 11 shows the results of adding gray scale to the faces of the model. On a color display, faces and parts of faces that are occluded in the original image are displayed in red. An interesting future problem involves incrementally updating the intensity patch of a face as information is acquired from successive images. Note that the gray scale displays might also be useful in performing a 2D match between the projected image of the model and an image of the real scene.

5. Multiple Views

This section describes an experiment in combining information from two views to generate the scene description. The 3D information shown in Fig. 9 is derived from one view (viewing the scene from the "front"). Another set of vertices and edges, depicted in Fig. 12, was manually generated to simulate the information available from an opposing point of view (viewing the scene from the "back"). The viewpoint for the perspective drawings of Figs. 9 and 12 are almost the same to allow easier comparison by the reader. Notice that the information in Fig. 9 emphasizes edges and vertices that are facing the front of the scene, while vertices and edges facing the back of the scene are emphasized in Fig. 12.

We have made the assumption in this experiment that we know the exact positions, relative to the first view, of the cameras used to obtain the second view. Therefore, the wire-frame descriptions in Figs. 9 and 12 can be expressed in the same coordinate system. We are currently working on the problem of matching such descriptions with a model so that relative positions of views can be automatically determined.

The procedure used in this experiment is similar to the one described in the last section, except that matching and merging of the two sets of wire-frames is also required.

First, for each set of wire frames, edges that are nearly parallel and very close to each other are merged. Next, each connected group of edges is labeled as a separate wire-frame object. We now want to merge objects derived from the first view with matching objects derived from the second view. Two objects are said to match if they have matching vertices or edges. The requirements for two vertices, one from each object, to match are: (1) they must be very close together, or (2) they must be part of matching edges, and the other two vertices of the edges match. The requirements for two edges, one from each object, to match are: (1) the two vertices of one edge must match the two of the other, or (2) one vertex of one edge matches one vertex of the other, and the two edges are close together and overlap in their lengths. These rules are used in a relaxation algorithm to obtain matching vertices and edges.

Two matching wire-frame objects are merged in the following manner. First, their matching vertices are merged. The coordinates of each resulting vertex are those of the midpoint of the line connecting the two initial vertices. Next, the matching edges are merged by using a type of "averaging" to obtain a resulting edge for two initial edges that do not coincide. This averaging is based on the observation that end points of edges that are vertices generally have much more accurate 3-space positions than end points that are not vertices. Therefore, the vertex end points are given greater weight in the averaging than the non-vertex end points. Finally, all other edges and vertices of the two objects are combined to generate a single wire-frame object.

From this point onward, processing continues as described in the previous section. Web faces are generated for each corner of each vertex, the web faces are merged, the shape of incomplete faces are completed, holes in faces are found, and vertical walls are dropped from faces floating above the ground. Fig. 13 shows several perspective views of the resulting scene model.

5.1. Results with Multiple Views

There are two important differences between the scene models shown in Figs. 13 and 10. First, the one in Fig. 13 contains more buildings. This is expected because more wire-frame data is available in constructing this model. Second, many buildings that are described in both models are more accurately described in the one in Fig. 13. That is, the positions of vertices and edges of these buildings are more precise. There are two reasons for this: (1) Since more *wire-frame* data is *available for reconstructing these* buildings, we obtain high accuracy for more vertices and edges. (2) Since many vertices and edges are redundantly available in both sets of data, their positions are "averaged", generally decreasing the amount of error.

This experiment demonstrates how the information provided by each additional view allows the scene model to be gradually refined and made more complete.

In this experiment, only wire-frame objects are matched and merged. Our next step will be to match and merge a wire-frame description with a scene model. Our current experiment can also be thought of as merging *wire frames* with a *scene model* by *noting* that it is equivalent to having generated a model from one set of wire frames, but using only confirmed vertices and edges of the model to match and merge with the other set of wire frames. This gives an indication of the importance of confirmed information for the more general matching and merging processes. Our next step will require determining which parts of the model, both confirmed and unconfirmed, require modification. Some of these parts may actually have to be pulled apart and rebuilt, while others may merely require modifications to their 3-space locations.

6. Conclusion

The current state of the 3D Mosaic project has been described. The goal of this project is to acquire a detailed 3D model of a complex scene from images. A useful approach to this problem is to acquire the model in an incremental manner, over a sequence of images taken from multiple viewpoints. We have also shown that task-specific knowledge is very useful in interpreting complex images. Our stereo analysis component uses such knowledge for matching features in the images, and our higher level reasoning component uses such knowledge for reconstructing shapes from the stereo output.

Fig. 14 displays a flow chart for the whole system. The stereo analysis extracts 3D wire-frame descriptions representing portions of boundaries of the buildings in the scene. A surface-based model representing an approximation of the scene is then generated from the wire-frame descriptions. This model should be useful for tasks such as matching, photointerpretation, display generation, and path planning. On a color display, the images in Fig. 11 would show red for parts of the scene not yet observed. This idea can be used in a task such as planning flight paths for reconnaissance, where a path that permits viewing the maximum amount of red portions might be optimal.

There are several extensions and improvements we have in mind for our system. In addition to continuing our experiments with multiple views as discussed in the previous section, the following are our main tasks for the immediate future:

1. Using the scene model for matching. This is required for performing model-based image understanding and for updating the model with information obtained from a new view.

2. Verifying a scene model in a top-down manner by projecting hypothesized edges and vertices into the image plane and then searching for them in the image.

3. Increasing the amount and accuracy of the wire-frame information extracted during stereo analysis. More boundaries of buildings in the scene than shown in Fig. 6 can probably be extracted by directly incorporating task-specific knowledge at the lowest levels in the process of extracting junctions from the image.

Acknowledgement

Fumi Komura did much work in exploring and experimenting with initial concepts dealing with this project. Duane Williams has provided excellent programming support and many ideas. In addition, Dave McKeown, Steve Shafer, and David Smith have provided useful comments and criticism.

References

1. Baer, A., Eastman, C., and Henrion, M. "Geometric Modelling: a Survey." *Computer-Aided Design 11* (September 1979).

2. Baker, H. H., and Binford, T. O. "Depth from Edge and Intensity Based Stereo." *Proc. IJCAI-81* (1981).

3. Barnard, S. T. and Thompson, W. B. "Disparity Analysis of Images." *IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-2*, 4 (July 1980).

4. Barrow, H. G., Bolles, R. C., Garvey, T. D., Kremers, J. H., Tenenbaum, J.M., and Wolf, H. C. "Experiments in Map-guided Photo Interpretation." *Proc. IJCAI-77* (August 1977).

5. Devich, R. N., and Weinhaus, F. M. "Image Perspective Transformations." *Proc. SPIE* (July 1980).

6. Doyle, J. "A Truth Maintenance System." *Artificial Intelligence 12* (1979), 231-272.

7. Duda, R.O. and Hart, P. E.. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, 1973.

8. Eastman, C. M., and Preiss, K. A Unified View of Solid Shape Modeling Based on Consistency Verification. Carnegie-Mellon University, September, 1981.

9. Hannah, M. J. Computer Matching of Areas in Stereo Images. Tech. Rept. AIM-239, Stanford University, July, 1974.

10. Henderson, R. L., Miller, W. J., and Grosch, C. B. "Automatic Stereo Reconstruction of Man-made Targets." *Proc. SPIE 186* (August 1979).

11. Liebes, S. "Geometric Constraints for Interpreting Images of Common Structural Elements: Orthogonal Trihedral Vertices." *Proc. Image Understanding Workshop* (April 1981).

12. Lucas, B. D., and Kanade, T. "An Iterative Image Registration Technique With an Application to Stereo Vision." *Proc. IJCAI-81* (August 1981).

13. Marr, D., and Poggio, T. "A Computational Theory of Human Stereo Vision." *Proc. R. Soc. Lond. B 204* (1979).

14. Nevatia, R. and Babu, K. R. An Edge Detection, Linking and Line Finding Program. Image Processing Institute, University of Southern California, September, 1978.

15. Rubin, S. *The ARGOS Image Understanding System*. Ph.D. Th., Carnegie-Mellon University, 1978.

16. Shafer, S. A., and Kanade, T. Using Shadows in Finding Surface Orientations. Tech. Rept. CMU-CS-82-100, Carnegie-Mellon University, January, 1982.

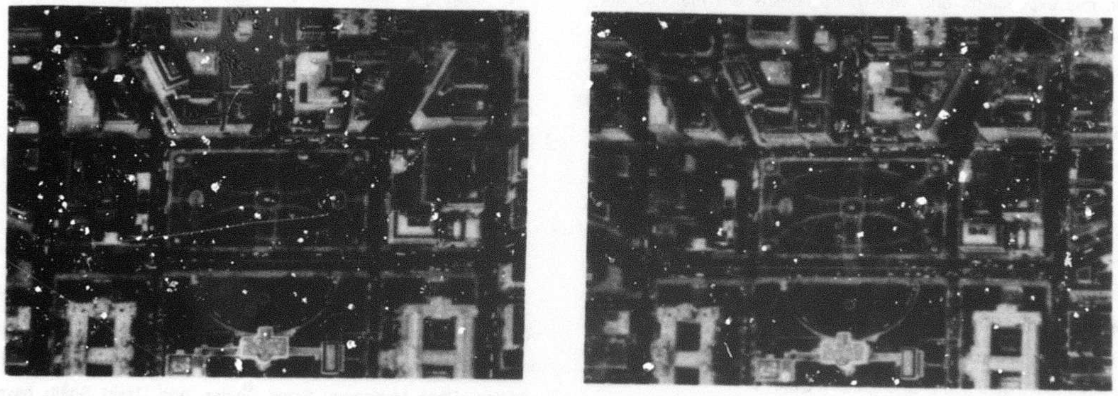


Figure 1: Gray scale stereo images of a region of Washington, D.C.

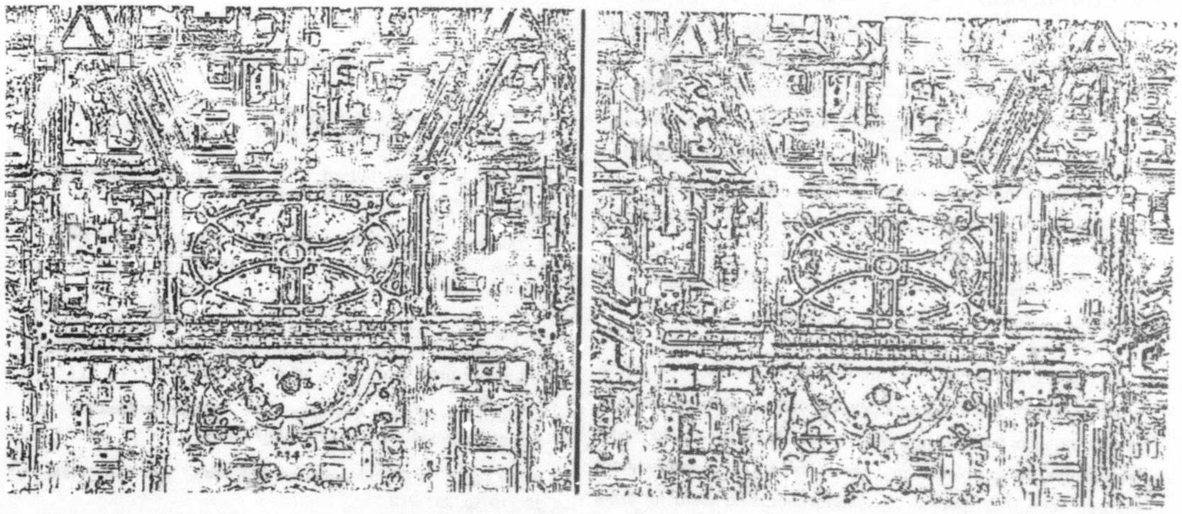


Figure 2: Edges resulting from a Sobel operator applied to the images of Fig. 1

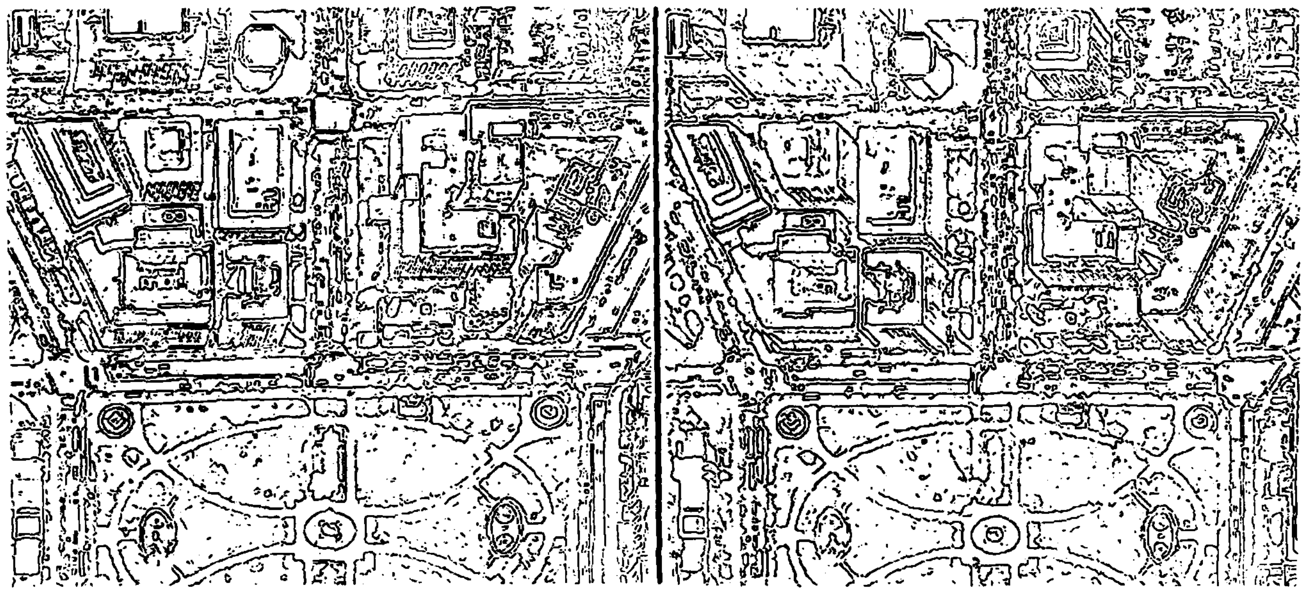


Figure 3: Result of thinning the edges in Fig. 2

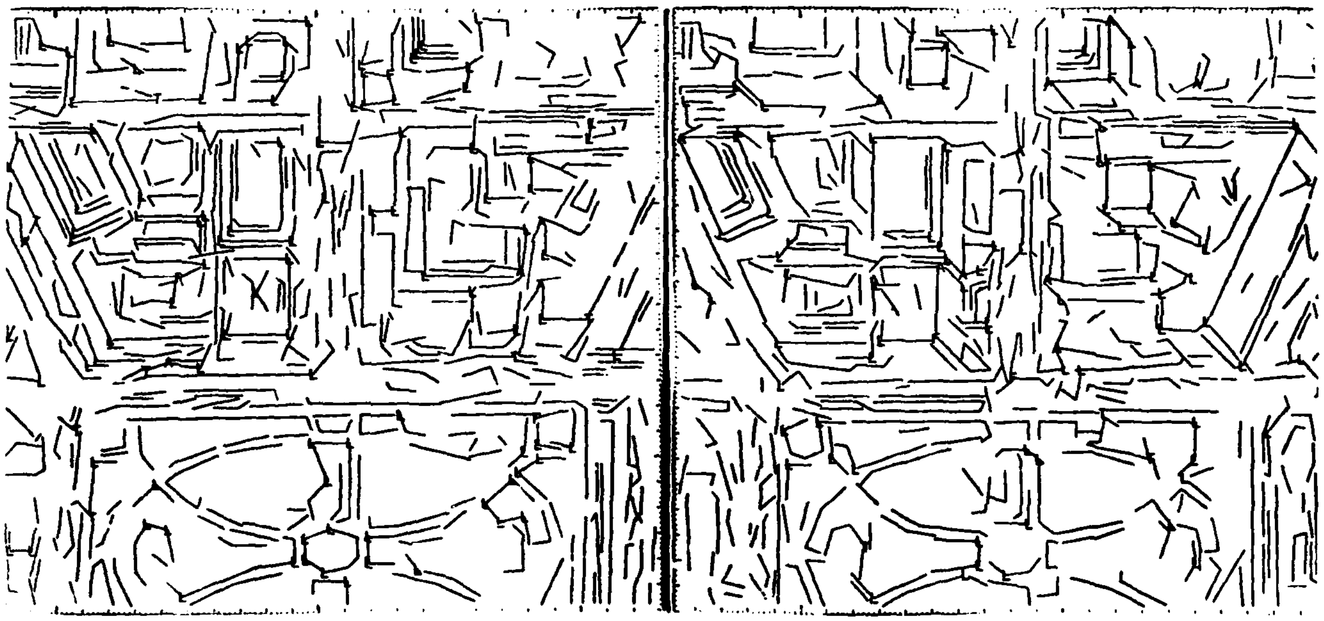


Figure 4: Straight lines fitted to the edge points of Fig. 3 after they are linked. Junctions in the images are classified as L, A (arrow), F (fork), or T.

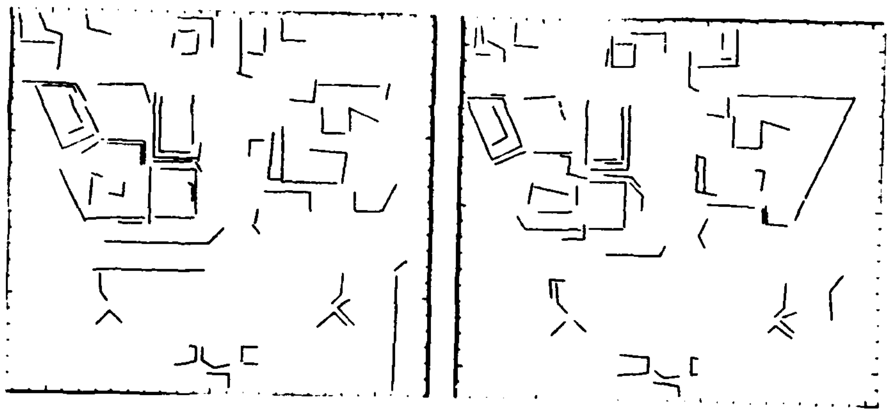


Figure 5: Matches that have been found for junctions and lines in the two images.

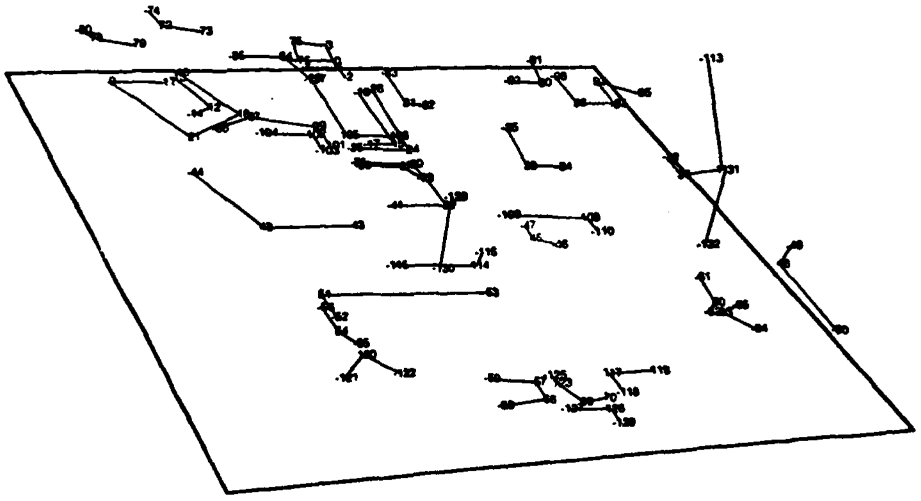


Figure 6: Three-dimensional perspective view of vertices and edges derived from matches shown in Fig. 5.

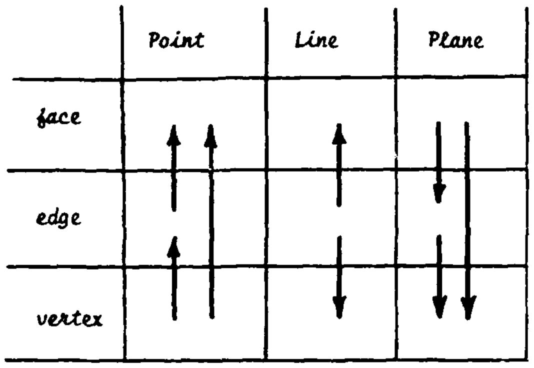
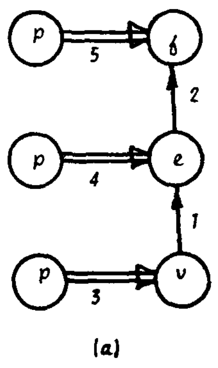


Figure 7: Rectangular boxes indicate geometric constraints on topological nodes. Arrows indicate direction of propagation of constraints.



v is part of *e* (link 1)
e is part of *f* (link 2)
p constrains *v* (link 3)
p constrains *e* (link 4)
p constrains *f* (link 5)

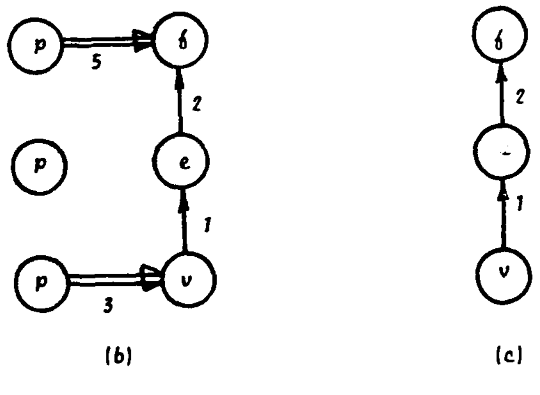


Figure 8: (a) Initial structure graph.(b) Link 4 is deleted.
(c) Resulting structure graph after effects of deletion have been propagated.

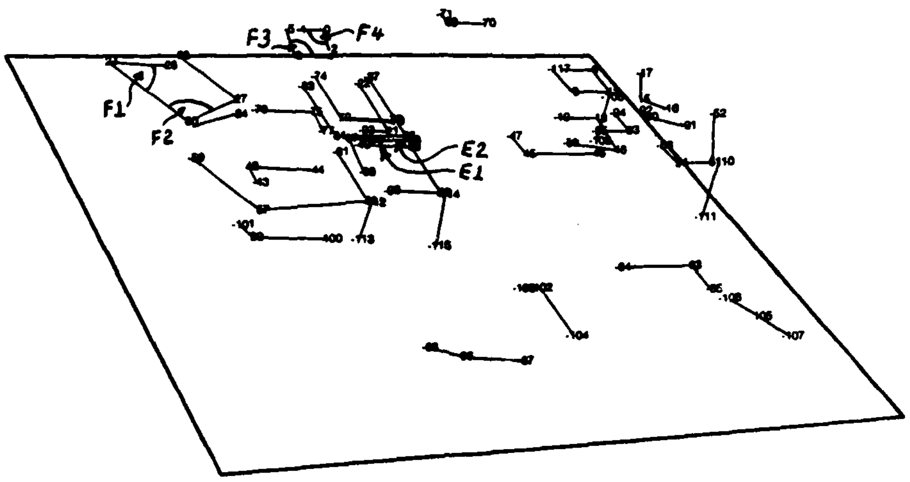


Figure 9: Three-dimensional perspective view of vertices and edges extracted from stereo pair. This version is different from the one shown in Fig. 6

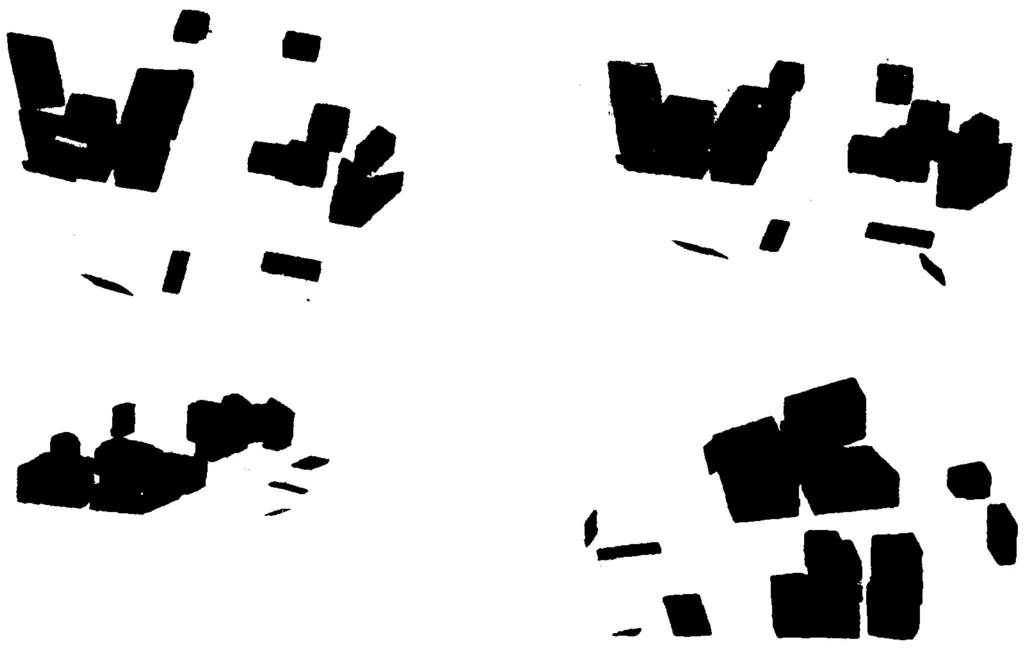


Figure 10: Three-dimensional perspective views of reconstructed buildings.

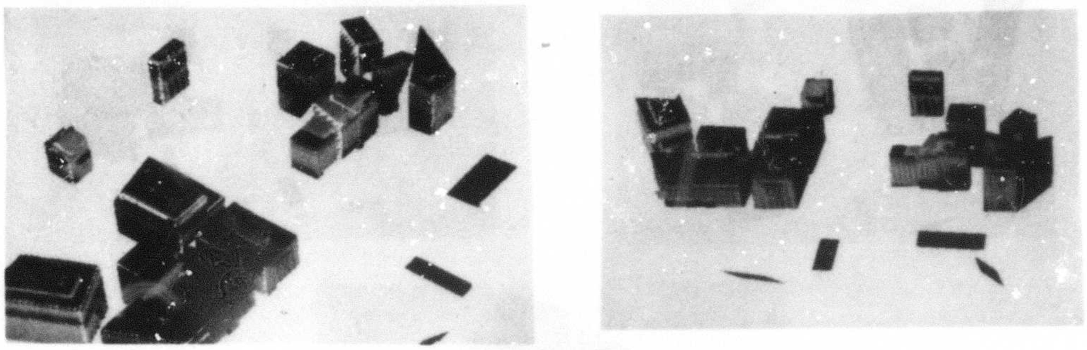


Figure 11: Reconstructed buildings with gray scale mapped onto faces. Gray scale values were derived from one of the images in Fig. 1. In a color display, faces and portions of faces that are occluded in the original image show up as red.

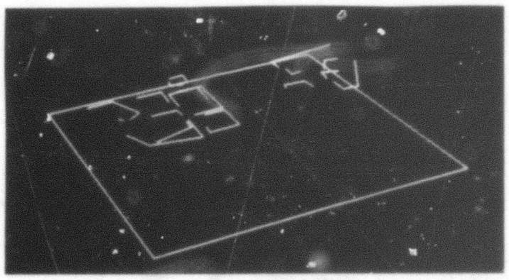


Figure 12: Three-dimensional perspective view of vertices and edges generated manually. This information might be derived from stereo analysis of images obtained from an opposite point of view from that shown in Fig. 1. The viewpoint for this perspective drawing is almost the same as for Fig. 9, to allow easier comparison by the reader.

Figure 13: Three-dimensional perspective views of buildings reconstructed from two views.

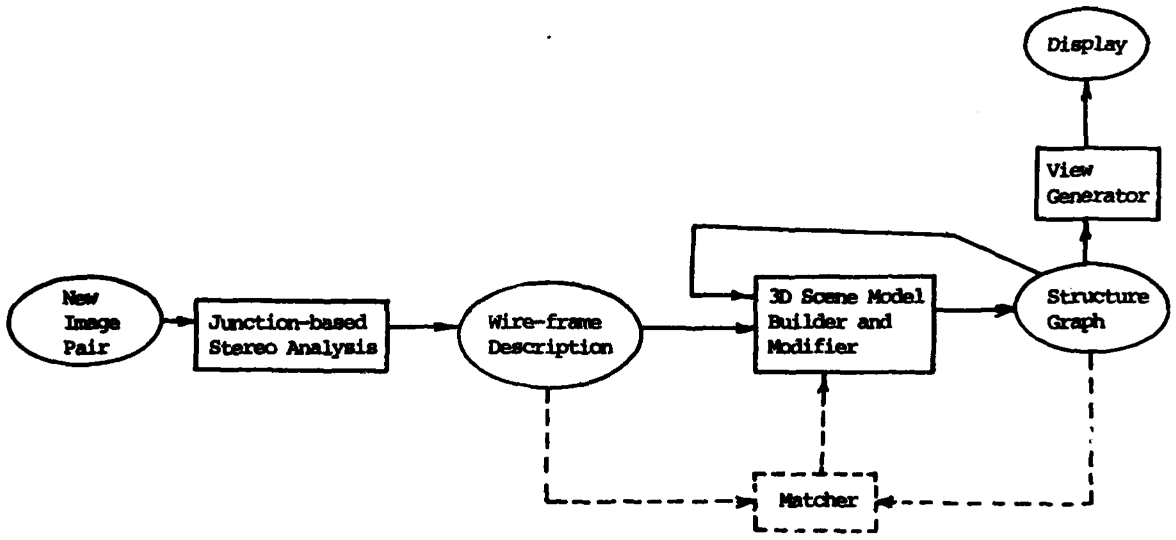


Figure 14: 3D Mosaic flowchart, showing major modules (boxes) and data structures (ellipses). The matcher has not yet been implemented.

AD-P000 125

METHODS FOR INTERPRETING PERSPECTIVE IMAGES.

Stephen T. Barnard

Artificial Intelligence Center, SRI International
Menlo Park, California 94025

ABSTRACT

A fundamental problem in computer vision is how, given an image, to determine the orientation of curves and surfaces in space. The problem is difficult because metric properties, such as orientation and length, are not invariant under projection. Under perspective projection (the correct model for most real images) the transform is nonlinear, and therefore hard to invert. Two methods are presented: one finds the orientation of parallel lines and planes by locating vanishing points and vanishing lines; the other determines the orientation of planes by backprojection of angles.

1 Introduction

A computational theory of vision must explain a very puzzling aspect of human visual experience. How is it that we correctly perceive three-dimensional properties of objects in space from two-dimensional projections (e.g., a single image)? At first, it seems that essential information for depth is lost when the retinal image is formed: a ray of light may as well have come from a star light-years distant as from across the room. Nevertheless, we have definite impressions of the distances and orientations of the things we see, even when there is no explicit, unambiguous information about these three-space relations in the image.

There are a few purely physical mechanisms that can account for some modes of spatial perception — in particular, accommodation of the lens to focus at different distances, binocular stereopsis, and optic flow. But while these mechanisms may account for some spatial perception, their explanation remains insufficient and incomplete. We usually have no trouble interpreting single images with substantial ranges of depth, or even simple line drawings with an infinite number of possible interpretations. Since information is lost in projecting a three-dimensional scene onto a two-dimensional surface, some form of computational "cognitive" model is required to construct percepts from ambiguous, incomplete, and noisy images.

Three important spatial properties that we perceive are **size** , **shape,** and **depth.** Size and shape are fundamentally different from depth because they are defined relative to an **object,** while depth is defined relative to an **observer.** Size is usually measured with ordinary Euclidean metrics: length, area, and volume. It is difficult to give a precise definition of shape, but the essential principle is that the shape of an object is the spatial arrangement of the contours and surfaces of which it is composed. While size is independent of the choice of a coordinate system, shape usually is not. Shape is often specified in some "natural" object-centered coordinate system that is selected and aligned to match the symmetry of the object.

In what follows, we shall assume that shapes can be adequately described by straight lines and planes. These primitive shape descriptors are the simplest geometrical contours and surfaces we can hope to find. They are common in scenes containing man-made objects, less common in natural scenes. If we can develop computational methods for the perception of lines and planes, we can perhaps generalize them to include more complex shapes.

To recover 3-D shape from 2-D projections, an explicit model of the projective transform is essential. Two models are common: parallel and central projection (Figure 1). In parallel projection an image is formed by parallel rays, usually perpendicular to the image plane. In central projection an image is formed by rays passing through a common point in space called the focal point. The parallel projective transform is called "orthographic," the central projective transform "perspective."

It is important to emphasize that central projection is the correct model both for human vision and for cameras, whereas parallel projection is only an approximation.

The most important parameter that distinguishes perspective from orthographic projection is the included angle of view, which is defined to be the maximum angle between two rays (i.e., the angle between the two rays with the greatest angular separation). The assumption of orthographic projection is essentially equivalent to the assumption of zero included angle of view. Locally, perspective projection is approximately orthographic because the included angle of view is small. When the entire image is considered, however, perspective is important.

If the focal length (the perpendicular distance from the focal point to the image plane) is large compared to the linear dimension of the image, the included angle of view is small and the orthographic approximation is reasonable. Photographs taken with "normal" lenses for a certain film format (e.g., a 50-mm lens on a 35-mm camera) typically cover about 45 degrees of view, and perspective effects are often quite apparent. If a wide-angle lens is used, perspective is dominant and the picture may appear distorted, although the "distortion" is merely the result of viewing the photograph from the wrong distance.

Another parameter that causes perspective images to differ from orthographic ones, even when the included angle of view is small, is the ratio of the distances to objects in the scene (or, informally stated, the ratios of "depths" of points). Under perspective, the projected area of an object varies inversely with the object's distance from the focal point (measured along the principal ray, defined to be the ray perpendicular to the image plane). Under orthography, however, the size of an object in an image is independent of depth.

The perspective camera model, therefore, will be required for accurate recovery of size, shape, and depth whenever the image covers a substantial included angle of view, or whenever objects at very different depths are compared.

The difference between orthographic and perspective projection is not only quantitative, but also qualitative. In Figure 2 one of the most familiar of all illusions — the Necker cube — is shown in parallel and central projection. In both cases the images are highly ambiguous because they could have been produced by an infinite number of objects; nevertheless, in each case we perceive only two distinct interpretations. The interpretations of the orthographic image are more or less equally preferable because both have the same symmetry. The interpretations of the perspective image, however, are radically different: one is a symmetrical cube and one is a relatively asymmetrical octohedron. There are other qualitative differences between orthography and perspective. For example, vanishing points and vanishing lines are not found in orthographic projections, but they are characteristic of perspective projections. (This topic will be covered in detail later.)

Attempts to use an explicit model of the projective transform have a long history in computer vision. Mackworth used the concept of **gradient space** [1], based on Huffman's **dual space** [2], to interpret line drawings of polyhedral scenes. Gradient space, combined with the parallel projection, is a useful tool because physical constraints on the scene can be represented as relations in gradient space. Horn, for example, used this approach in his analysis of "shape-from-shading" [3]. An overview of gradient-space methods can be found in [4].

For reasons that will be made clear in Section 2, perspective involves more difficult mathematics than does orthography. (See Haralick for a discussion of the mathematics of perspective [5].) Most computer vision approaches, therefore, begin with the assumption of parallel projection. One notable exception is Kender [6], who argued that gradient space is not an ideal domain for representing constraints under perspective. A different domain — the Gaussian sphere, which will be described in Section 3 — is much more useful.

2 Mathematics of Perspective

2.1 Algebraic Methods

In this section the basic mathematical models of central projection will be reviewed.

Central projection can be represented as a simple linear transform. Given a point $\mathbf{p} = (x, y, z)$, the parallel projection $\mathbf{p}'$ of $\mathbf{p}$ is given by:

$$\mathbf{p}' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{p}^T . \tag{2.1}$$

Central projection, on the other hand, is an essentially nonlinear transform: image coordinates are determined by dividing scene coordinates by the depth as measured along the principal ray. The central projection $\mathbf{p}'$ of $\mathbf{p}$ onto an image plane at $z = 0$ through a focal point at $(0, 0, -f)$ is given by

$$\mathbf{p}' = \left(\frac{xf}{z+f}, \frac{yf}{z+f}, 0\right) . \tag{2.2}$$

The perspective transform can be expressed elegantly with homogeneous coordinates. Homogeneous coordinates were first developed as an analytical tool for projective geometry [7]; more recently they have been used effectively in computer graphics [8] and industrial automation [9]. The homogeneous coordinates of a point are represented by a four-tuple (x, y, z, w), and the ordinary three-dimensional coordinates of the point are obtained by $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})$. One advantage in using homogeneous coordinates in projective geometry derives from the fact that points "at infinity" are represented as four-tuples with $w = 0$, whereas in ordinary coordinates they have no representation.

The formulation of central projection in homogeneous coordinates is as follows: a point $\mathbf{P}$ with homogeneous coordinates (x, y, z, w) is projected onto the image plane at point $\mathbf{q}$ with ordinary coordinates $\mathbf{q} = (\frac{xf}{z+fw}, \frac{yf}{z+fw}, 0)$. This projection can be expressed in matrix form as

$$\begin{aligned} \mathbf{P}' &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{pmatrix} \mathbf{P}^T \\ &= (x, y, z, \frac{z+wf}{f}) , \end{aligned} \tag{2.3}$$

followed by conversion to ordinary coordinates

$$\mathbf{p}' = \left(\frac{xf}{z+wf}, \frac{yf}{z+wf}, \frac{zf}{z+wf}\right) , \tag{2.4}$$

and finally a "parallel" projection transform

$$\mathbf{q} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{p}'^T . \tag{2.5}$$

Clearly, parallel projection is a special case of central projection. If the focal length is infinite Eq. (2.3) becomes the identity transform.

2.2 A Computational Approach

Geometric properties can be divided into two classes: metric, such as the length and orientation of lines; and descriptive, such as the colinearity of three points or the coincidence of three lines. Metric properties are in general not invariant under projection (either parallel or central), but descriptive ones are. One general approach to using a camera model for image interpretation (and the one used here) is to first identify instances of descriptive attributes in the image, which, since they are invariant under projection, express strong, unambiguous, and often global information about the scene. These descriptive attributes can then be combined with geometric constraints (the camera model), with heuristic rules (such as a preference for symmetrical figures), and with specific knowledge of the scene to infer metric properties.

Under orthographic projection the parallelism of lines (a descriptive property) is invariant, but under perspective projection it is not, and must therefore be replaced with a more general property. A central projection maps parallel lines in space onto a pencil of lines intersecting at a common point on the image plane. This point of intersection, called a vanishing point, has important implications for image interpretation. In perspective, consequently, parallelism as a descriptive property is replaced by "coincidence." In Section 3 a computational method for finding vanishing points is described.

Under both orthographic and central projection the metric properties of angles are transformed in highly ambiguous ways. An angle on a plane in three-space (defined as the intersection of two lines in the plane) can project to any angle in the image, depending on the orientation of the plane with respect to the image. From a more optimistic standpoint, we can say that angles in the image constrain the orientation of planes in three-space. In Section 4 an algorithm is presented for finding the orientation of planes from image angles and heuristic symmetry assumptions. Computational methods for shape perception sometimes exploit known or suspected symmetry in the scene to decide among multiple interpretations. Kanade has used this approach for interpreting orthographic projections [10].

3 Vanishing Points

This section describes a method for finding vanishing points in a perspective image and discusses how to use them to interpret the scene.

The approach is based on the assumption that there exist groups of parallel straight structures in the scene, and that these structures produce line segments in the image. According to the laws of perspective, such a group of image line segments, when extended, will intersect at a common vanishing point. This point has the following interpretation: it is the projection of the intersection of the parallel lines "at infinity." Once the vanishing point is located, the orientation of the group of parallel three-space lines is established (assuming that the focal length is known). This is illustrated in Figure 3.

The problem of finding vanishing points is divided into (1) finding line segments in the image and (2) finding intersections of the extended line segments that are likely to be vanishing points.

Problem (1) is solved with well-known, conventional techniques that will not be discussed here.

Problem (2), that of finding intersections, is greatly simplified by using a transform called Gaussian mapping [11]. The problem with trying to find intersections directly in the image is that the image plane is an open space, and the vanishing points may occur anywhere, even "at infinity." (The use of gradient space to represent surface orientation raises the same problem: as the surface normal approaches 90 degrees from the z-axis, the gradient-space point approaches infinity.)

Gaussian mapping transforms vectors in three-space into points on a unit sphere at the origin (Figure 4). The vectors can represent either planar normals or the direction cosines of lines. Since all parallel vectors in space map to the same point on the sphere, any point on the sphere represents a family of parallel vectors. There is an interesting dual relationship between lines and planes in projective space: two lines determine a plane, and two planes determine a line. There is a similar dual relationship on the Gaussian sphere between points and lines (i.e., great circles). A point on the sphere determines a pole through the origin; the dual of the point is the equator associated with the pole.

The **interpretation plane** associated with an image line is defined as follows (Figure 5). Let $\mathbf{p}_1 = (x_1, y_1, f)$ and $\mathbf{p}_2 = (x_2, y_2, f)$ be two distinct image points defining a line l. Then the interpretation plane ϕ associated with l is the plane containing l and the origin (i.e., the focal point), and can be represented by its unit normal:

$$\phi = \frac{\mathbf{p}_1 \times \mathbf{p}_2}{|\mathbf{p}_1||\mathbf{p}_2|} . \qquad (3.1)$$

The plane ϕ is called the interpretation plane of l because the line in space, the projection of which l, must lie in ϕ.

The interpretation planes of image lines intersect the Gaussian sphere in great circles, as shown in Figure 6. The intersections of these great circles on the sphere correspond exactly to intersections of their associated lines in the image plane. The procedure for finding vanishing points is then as follows: (1) find lines in the image; (2) determine the interpretation plane of each line; (3) trace the great-circle intersections of the interpretation planes with the Gaussian sphere; (4) find the points on the sphere where several great circles intersect. The vanishing points can be projected back onto the image plane, if desired.

After a vanishing point has been found, its dual interpretation can also be very useful for interpreting the image. The dual of a vanishing point is a **vanishing line** (a great circle on the sphere). For example, if the vertical vanishing point is found, its dual is the "horizon line" (i.e., the vanishing line of all horizontal planes). As another example, if two horizontal vanishing points are found, their duals intersect at the vertical vanishing point.

The Gaussian sphere can be digitally represented as a two-dimensional array of real numbers, with the row index indicating an azimuth and a column index representing an elevation. Each array element corresponds to a small surface area of the sphere. (In this representation the surface areas are not equal.

Other forms, such as tessellated regular polyhedra, might be better suited to representing the sphere, but they are rather complicated to implement.)

The procedure for tracing a great circle in the sphere array is as follows. Let α and β be the spherical coordinates of a point, where α is the azimuth measured from the z-axis and β is the elevation measured from the xy-plane. In Cartesian coordinates, the point is

$$\mathbf{g} = (\sin\alpha\cos\beta, \sin\beta, \cos\alpha\cos\beta) \ . \tag{3.2}$$

The equation of the great circle associated with interpretation plane $\phi = (\phi_x, \phi_y, \phi_z)$ is

$$\mathbf{g}\cdot\phi = 0 \ , \tag{3.3}$$

from which we can derive

$$\beta(\alpha,\phi) = \tan^{-1}\frac{-\phi_x\sin\alpha - \phi_z\cos\alpha}{\phi_y} \ . \tag{3.4}$$

If ϕ_y is small, this equation can be replaced by a slightly different form that gives azimuth as a function of elevation and has ϕ_x in the denominator.

The array is first initialized to zeros. Since we have the elevation β as a function of the azimuth α and an interpretation plane ϕ (3.4), we can generate the great circle of ϕ in the array. When a curve is traced into the array, a real value associated with the curve is added to all the array elements containing the curve. (This value is derived heuristically from the length and goodness-of-fit of the image line.) Points where many curves intersect form clusters of high values; these indicate likely vanishing points.

Figures 7 and 8 show two examples of the method's application to real images. They were recorded onto 50 mm x 50 mm film areas with a 50-mm-focal-length lens (considered "wide-angle" for this film format) and were digitized at a resolution of 100 microns/pixel.

In Figure 7c, only the vertical vanishing point is clearly found on the Gaussian sphere. In Figure 7d, the dual of the vertical vanishing point is shown to be the horizon line. In Figure 7e, those image line segments whose interpretation planes contain the vertical vanishing point are shown. In Figure 8 two horizontal vanishing points are found; the vertical vanishing point is located by intersecting their duals.

4 Orientation of Planes

In this section, we shall consider a somewhat different algorithm that also uses Gaussian mapping to deal with the perspective camera model. The object of the algorithm is to find the orientation of a plane in three-space from a perspective image of a line figure drawn on the plane. Heuristic assumptions about the figure will be used constrain the orientation of the plane.

Assume that the line figure is a closed planar polygon with nonintersecting sides, such as a triangle. Each pair of adjacent sides defines an angle, and each angle in the planar figure generally projects to a different angle in the image (Figure 9). (The metric property of angle size is not preserved under projection.) Nevertheless, an angle measured in the image constrains the angle measured in the planar figure, but the constraint ranges over a family of possible planar orientations. In essence, the angle in the image can "backproject" onto any plane in space.

Consider two interpretation planes, $\phi_1 = (\phi_{1_x}, \phi_{1_y}, \phi_{1_z})$ and $\phi_2 = (\phi_{2_x}, \phi_{2_y}, \phi_{2_z})$, associated with two image lines that make angle ω in the image plane $\gamma = (0,0,-1)$.

$$(\gamma\times\phi_1)\cdot(\gamma\times\phi_2) = |\gamma\times\phi_1||\gamma\times\phi_2|\cos\omega \ . \tag{4.1}$$

or, substituting for γ,

$$\cos\omega = \frac{\phi_{1_x}\phi_{2_x} + \phi_{1_y}\phi_{2_y}}{\sqrt{\phi_{1_x}^2 + \phi_{1_y}^2}\sqrt{\phi_{2_x}^2 + \phi_{2_y}^2}} \ . \tag{4.2}$$

The angle ϕ_1 and ϕ_2 make on a plane γ' that has an arbitrary orientation of α in azimuth and β in elevation can be found by rotating the image plane γ in azimuth and elevation, and then evaluating Eq. (4.1). We actually use a slightly different approach, rotating ϕ_1 and ϕ_2 by $-\alpha$ in azimuth and $-\beta$ in elevation, and then evaluating Eq. (4.2) directly. This rotation of an interpretation plane ϕ into plane ϕ' is given by

$$\phi' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{pmatrix}\begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}\phi \ . \tag{4.3}$$

Using (4.2) and (4.3), projected angles measured in the image can be expressed as constraints on the orientation of the plane containing the angle in space.

One approach to using the backprojection constraint would be to solve a system of equations in the form of (4.2), but such an explicit solution appears to be very difficult. Instead, using a highly parallel algorithm, we backproject each image angle onto planes of all possible orientations (subject to some discrete quantification of the Gaussian sphere), obtaining at each orientation a value that expresses the angle the figure must make on such a plane.

Two examples of this method are illustrated in Figures 10 and 11.

The triangle in Figure 10a is interpreted as an image of some other triangle in space. Each of the three angles is backprojected onto planes of all possible orientations (Figure 10b). Each image in Figure 10b represents a map of the back of the Gaussian sphere. Each point represents a possible planar orientation in terms of the Gaussian spherical coordinates of the plane's normal. The intensity value at each point is directly related to $\cos\omega$ for that orientation (e.g., black indicates $\cos\pi = -1$ and white indicates $\cos 0 = 1$).

Knowledge or heuristic assumptions about the values of angles in space can be used to choose particular interpretations of planar orientation. For example, suppose we interpret the triangle as being as symmetrical as possible -- namely, an equilateral triangle ($\omega = 60$ degrees). Contours for this value of omega are shown in the Figure 10b. Note that the triangle yields two solutions. This result is consistent with human perception of the figures.

Similarly, the quadrangle in Figure 11a is interpreted as an image of some other quadrangle in space. In this case four angles are backprojected (Figure 11b). If we assume the quadrangle in space is a square ($\omega = 90$ degrees), and plot the contours for this value (shown in Figure 11b), we find that it yields a unique solution.

5 Conclusions

The perspective camera model is crucial for the interpretation of real images. Although parallel projection provides an adequate approximation when the included angle of view and the range of depth in the scene are small, these conditions are never completely satisfied. Perspective camera modeling entails more difficult mathematics than does orthography, but it also provides more powerful aids to perception (e.g., the Necker cube example in Section 1).

Gaussian mapping is a useful tool for the analysis of perspective images. In Section 3, Gaussian mapping was used to identify descriptive geometric properties (the coincidence of parallel lines), and to infer metric properties (the orientation of groups of parallel lines). The dual interpretation of vanishing points on the Gaussian sphere was used to extend the analysis to finding vanishing lines.

In Section 4, we described the technique of backprojecting of angles in the image to constrain the interpretation of planes in space. Once again the Gaussian sphere was used to represent the space of possible interpretations. Assumptions about the symmetry of figures in space, combined with the constraint surfaces obtained through backprojection, resulted in quantitative measurement of the orientation of the figures.

ACKNOWLEDGMENT

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract MDA903-79-C-0588. This contract is monitored by the U.S. Army Engineer Topographic Laboratory.

REFERENCES

1. Mackworth, A.K., Interpreting picture of polyhedral scenes, *Artificial Intelligence* **4** (1973) 121-137.
2. Huffman, D.A., Impossible objects as nonsense sentences, in: Meltzer, B. and Mitchie, D., (Eds.), *Machine Intelligence*, **6** (Edinburgh University Press, Edinburgh, 1971).
3. Horn, B.K.P., Obtaining shape from shading information, in: Winston, P.H., (Ed.), *The Psychology of Computer Vision*, (McGraw-Hill, New York, 1975).
4. Shafer, S.A., Kanade, T., and Kender, J., Gradient space *under orthography and perspective*, Computer Science Department, Carnegie-Mellon University, CMU-CS-82-123, 1982.
5. Haralick, R.M. Using perspective transformation in scene analysis, *Computer Graphics and Image Processing*, **13** (1980) 191-221.
6. Kender, J., *Shape From Texture*, Ph.D. Thesis, Carnegie-Mellon University, Computer Science Department, November, 1980.
7. Coxeter, H.S.M. *Introduction to Geometry*, (John Wiley & Sons, Inc., New York, 1961).
8. Newmann, W.M. and Sproull, R.F., *Principles of Interactive Computer Graphics*, 2nd. edition, (McGraw-Hill, New York 1979).
9. Paul, R.P., *Robot Manipulators*, (The MIT Press, Cambridge, Massachusetts, 1982).
10. Kanade, T., Recovery of the 3-D shape of an object from a single view, *Artificial Intelligence* **17** (1981) 409-460.
11. Hilbert, D. and Cohn-Vossen, S., *Geometry and the Imagination* (Chelsea Publishing Co., New York, 1952).

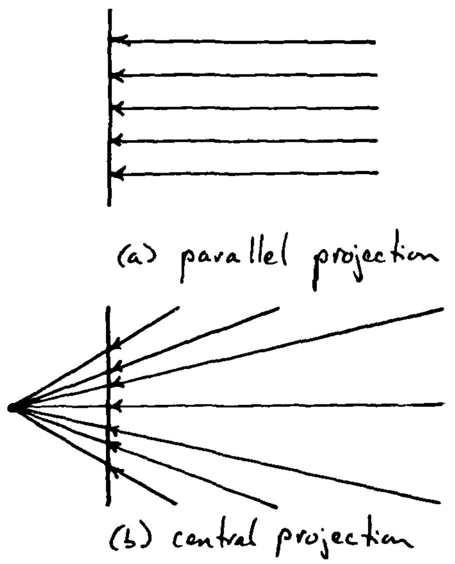


Figure 1 Parallel and central projection.

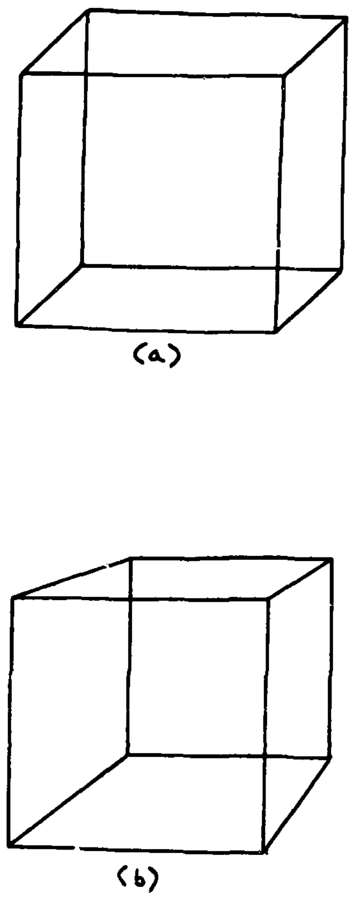


Figure 2 Necker cube illusion: (a) orthographic projection, (b) perspective projection.

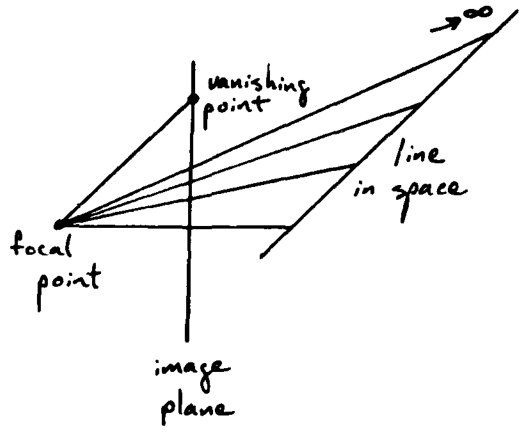


Figure 3 Construction of a vanishing point.

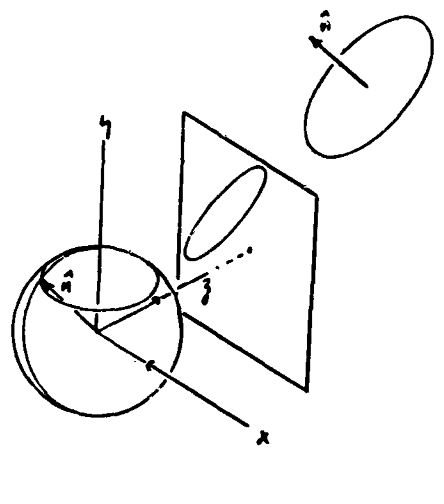


Figure 4 Gaussian mapping.

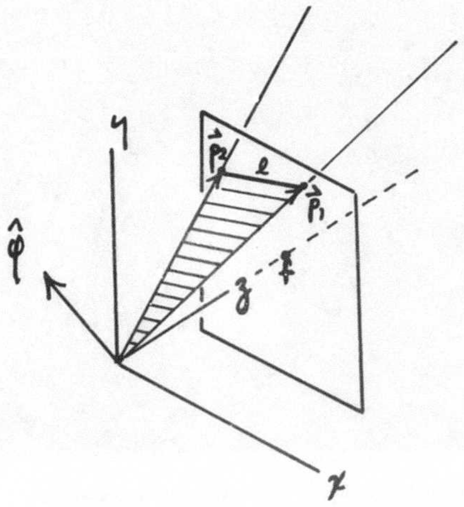


Figure 5 The interpretation plane.

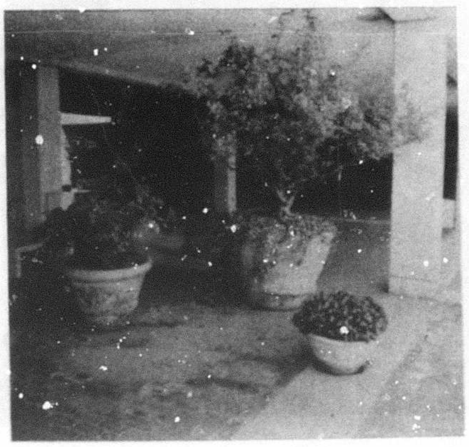


Figure 7a) Original image.

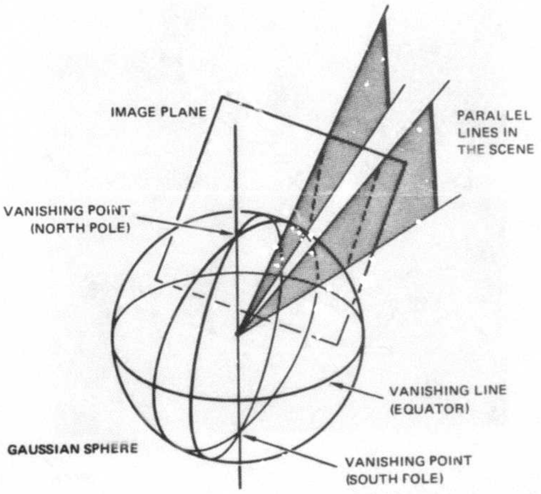


IMAGE PLANE
PARALLEL LINES IN THE SCENE
VANISHING POINT (NORTH POLE)
VANISHING LINE (EQUATOR)
GAUSSIAN SPHERE
VANISHING POINT (SOUTH POLE)

Figure 6 Vanishing points on the Gaussian sphere.

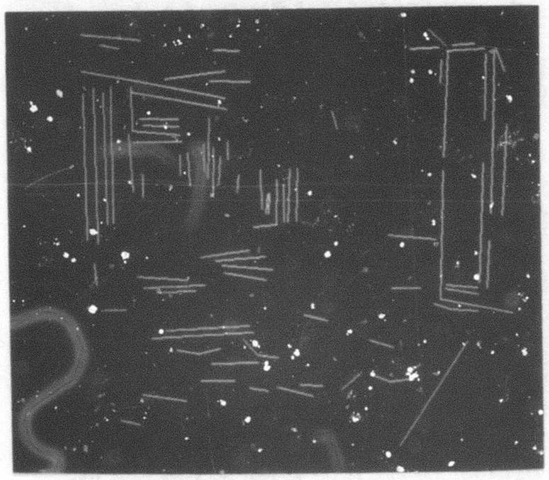


Figure 7b) Line segments.

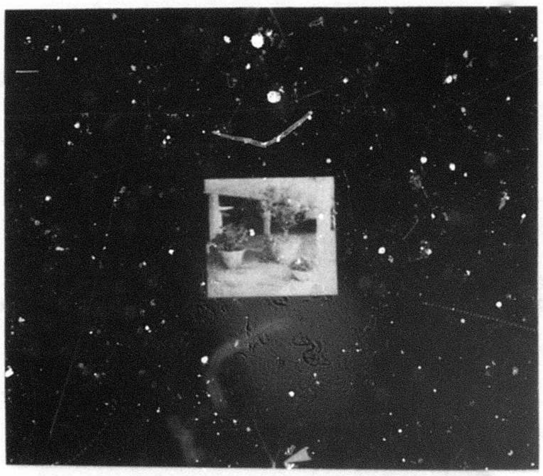


Figure 7c) Lines mapped onto the Gaussian sphere (with a vertical vanishing point).

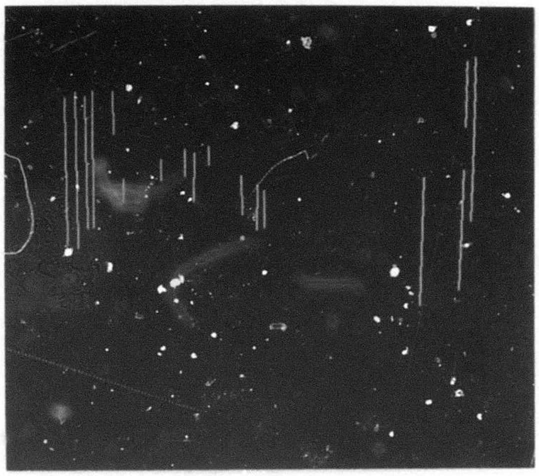


Figure 7e) Line segments whose interpretation planes contain the vanishing point.

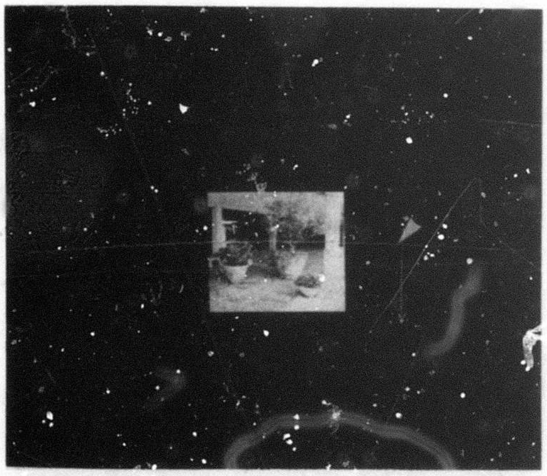


Figure 7d) The dual of the vertical vanishing point (the horizon line).

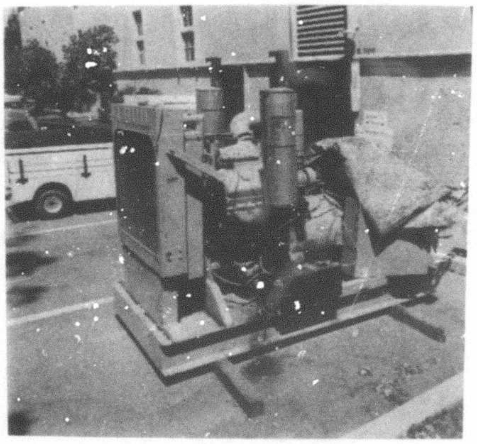


Figure 8a) Original image.

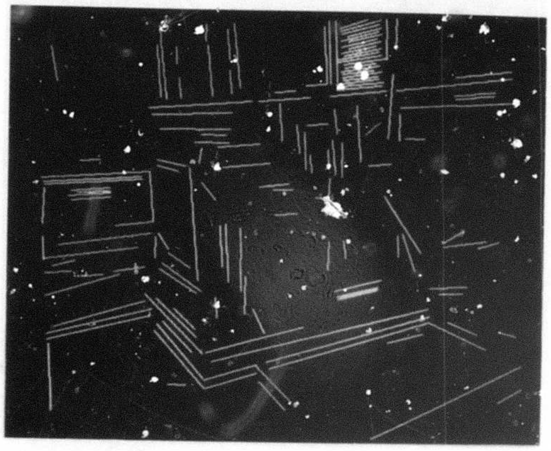


Figure 8b) Line segments.

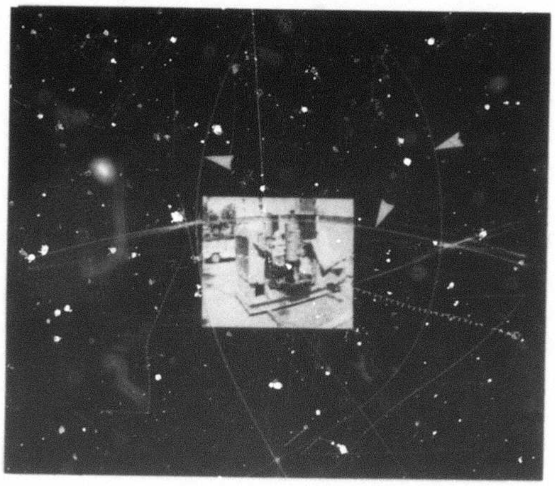


Figure 8d) The duals of the horisontal vanishing points, which intersect at the vertical vanishing point.

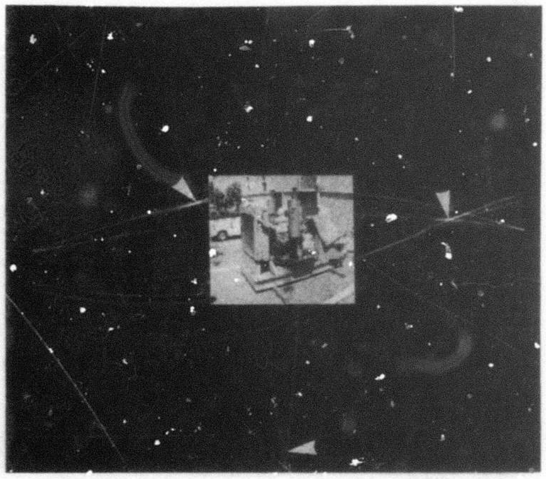


Figure 8c) Lines mapped onto the Gaussian sphere (with a vertical and two horisontal vanishing points).

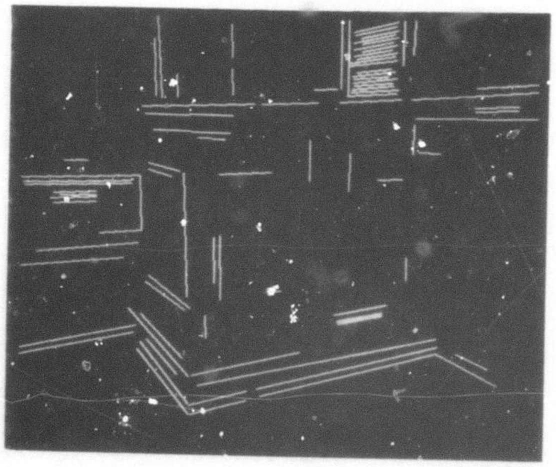


Figure 8e) Line segments whose interpretation planes contain the vanishing point.

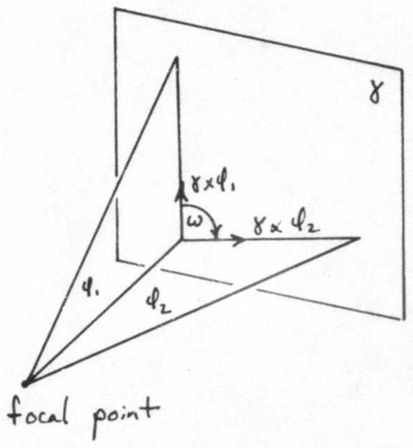


Figure 9 The angle two interpretation planes make on another plane in space.

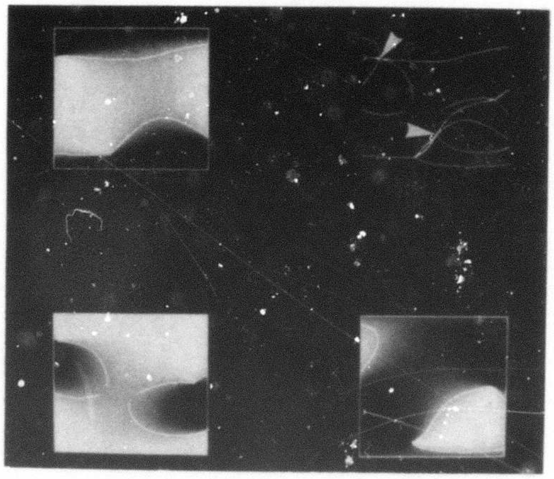


Figure 10b) Backprojection of angles onto planes of all orientations, showing contours for angles of 60 degrees (two planar orientations are possible).

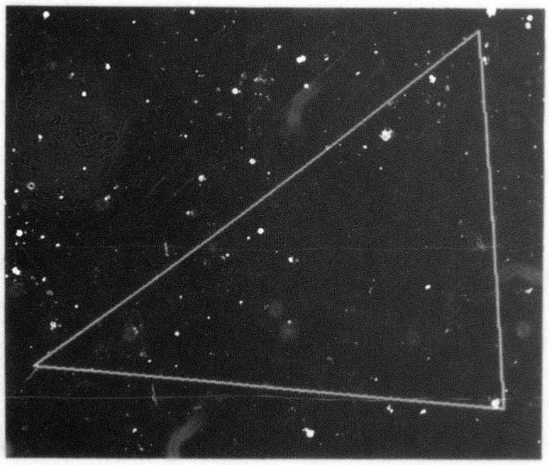


Figure 10a) A triangle.

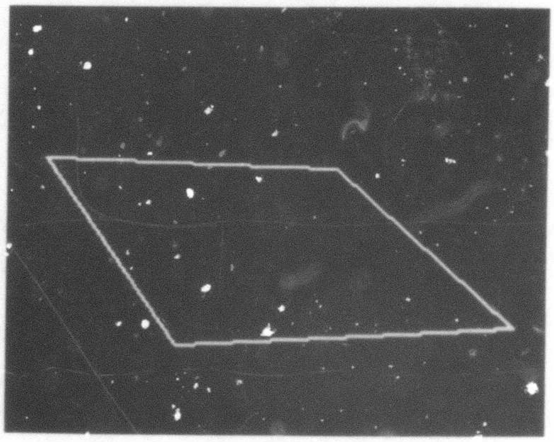


Figure 11a) A quadrangle.

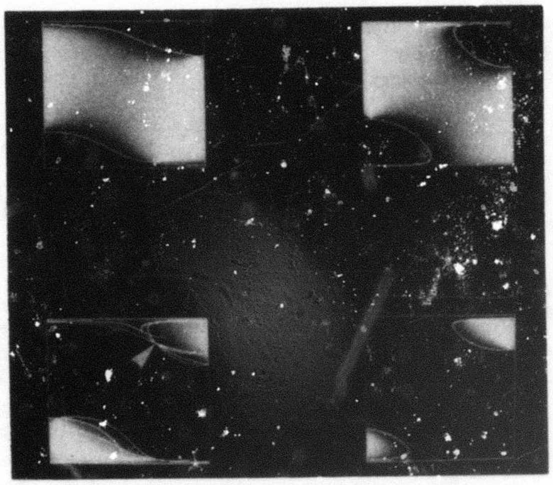


Figure 12b) Backprojection of angles onto planes of all orientations, showing contours for angles of 90 degrees (only one planar orientation is possible).

AD-P000 126

PASSIVE NAVIGATION

Anna R. Bruss * and Berthold K.P. Horn
Artificial Intelligence Laboratory
Massachusetts Institute of Technology

* Authors present address:

IBM, T.J. Watson Research Center
P. O. Box 218
Yorktown Heights, N. Y. 10598

Abstract

A method is proposed for determining the motion of a body relative to a fixed environment using the changing image seen by a camera attached to the body. The optical flow in the image plane is the input, while the instantaneous rotation and translation of the body are the output. If optical flow could be determined precisely, it would only have to be known at a few places to compute the parameters of the motion. In practice, however, the measured optical flow will be somewhat inaccurate. It is therefore advantageous to consider methods which use as much of the available information as possible. We employ a least-squares approach which minimizes some measure of the discrepancy between the measured flow and that predicted from the computed motion parameters. Several different error norms are investigated. In general, our algorithm leads to a system of nonlinear equations from which the motion parameters may be computed numerically. However, in the special cases where the motion of the camera is purely translational or purely rotational, use of the appropriate norm leads to a system of equations from which these parameters can be determined in closed form.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643 and in part by National Science Foundation Grant MCS77-07569.

1. Introduction

In this paper we investigate the problem of passive navigation using optical flow information. Suppose we are viewing a film. We wish to determine the motion of the camera from the sequence of images, assuming that the instantaneous velocity of the brightness patterns, also called the *optical flow*, is known at each point in the image. Several schemata for computing optical flow have been suggested (e.g. [2], [3], [5]). Other papers (e.g. [9], [11], [12]) have previously addressed the problem of passive navigation. Three approaches can be taken towards a solution which we term the *discrete*, the *differential* and the *continuous* approach.

In the discrete approach, information about the movement of brightness patterns at only a *few* points is used to determine the motion of the camera. In particular, using such an approach, one attempts to identify and match discrete points in a sequence of images. Of interest in this case is the photogrammetric problem of determining what the minimum number of points is from which the motion can be calculated for a given number of images [10, [11], [12], [16], [17]. This approach requires that one tracks features, or identifies corresponding features in images taken at different times. In their work, Tsai and Hunag [16] assumed that such corresponding points can be determined in two image. Then they showed that in general seven points are sufficient to determine the motion uniquely. They prove furthermore that such points have to satisfy a fairly weak constraint. Longuet-Higgins work [10] is fairly similar to [16] but he fails to show under which conditions the motion can be determined *uniquely* from corresponding points.

In the differential approach, the first and second spatial partial derivatives of the optical flow are used to compute the motion of a camera [6], [9]. It has been claimed that it is sufficient to know the optical flow and both its first and second derivatives at a single point to uniquely determine the motion [9]. This is incorrect (except for a special case) [1]. Furthermore, noise in the measured optical flow is accentuated by differentiation.

In the continuous approach, the whole optical flow field is used. A major shortcoming of both the local and differential approaches is that neither allows for errors in the optical flow data. This is why we choose the continuous approach and devise a least-squares technique to determine the motion of the camera from the measured optical flow.

The proposed algorithm takes the abundance of available data into account and is robust enough to allow numerical implementation.

Independently, Prazdny chose in [13] a similar approach to ours. He also proposes the use of a least-squares method to determine the motion parameters but never discusses how exactly this is to be done. Consequently, he does not show whether his scheme can be used to *uniquely* determine the motion.

2. Technical Prerequisites

In this section we review the equations describing the relation between the motion of a camera and the optical flow generated. We use essentially the same notation as [9]. A camera is assumed to move through a static environment. Let a coordinate system X, Y, Z be fixed with respect to the camera, with the Z-axis pointing along the optical axis. If we wish, we can think of the environment as moving in relation to this coordinate system. Any rigid body motion can be resolved into two factors, a translation and a rotation. We will denote by $\vec{T}$ the translational component of the motion of the camera and by $\vec{\omega}$ its angular velocity (see also Figure 1 which is redrawn from [9]). Let the instantaneous coordinates of a point P in the environment be (X,Y,Z).

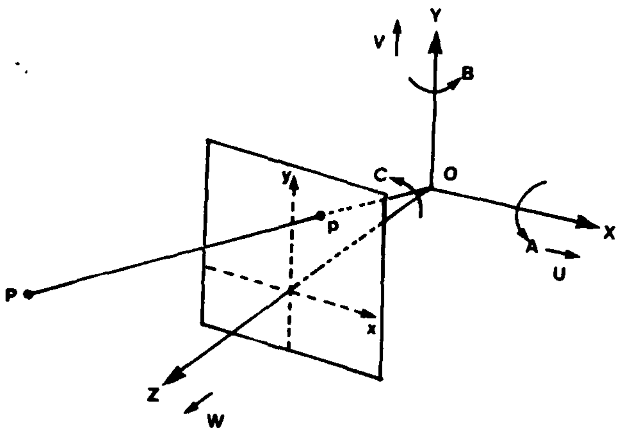


Figure 1. Coordinate Systems

(Note that $Z > 0$ for points in front of the imaging system.) Let $\vec{r}$ be the vector $(X,Y,Z)^T$, where ^{T} denotes the transpose of a vector, then the velocity of P with respect to the X, Y, Z coordinate system, is:

$$\vec{V} = -\vec{T} - \vec{\omega} \times \vec{r}. \tag{1}$$

We define the components of $\vec{T}$ and $\vec{\omega}$ as:

$$\vec{T} = (U,V,W)^T \qquad \vec{\omega} = (A,B,C)^T. \tag{2}$$

Thus we can rewrite (1) in component form:

$$\begin{aligned} X' &= -U - BZ + CY \\ Y' &= -V - CX + AZ \\ Z' &= -W - AY + BX. \end{aligned} \tag{3}$$

where $'$ denotes differentiation with respect to time.

The *optical flow* at each point in the image plane is the instantaneous velocity of the brightness pattern at that point. Let (x,y) denote the coordinates of a point in the image plane (see Figure 1). Since we assume perspective projection between an object point P and the corresponding image point p, the coordinates of p are:

$$x = \frac{X}{Z} \qquad y = \frac{Y}{Z}. \tag{4}$$

The optical flow, denoted by (u,v), at a point (x,y) is:

$$u = x' \qquad v = y'. \tag{5}$$

Differentiating (4) with respect to time and using (3) we obtain the following equations for the optical flow:

$$\begin{aligned} u &= \frac{X'}{Z} - \frac{XZ'}{Z^2} \\ &= (-\frac{U}{Z} - B + Cy) - x(-\frac{W}{Z} - Ay + Bx) \\ v &= \frac{Y'}{Z} - \frac{YZ'}{Z^2} \\ &= (-\frac{V}{Z} - Cx + A) - y(-\frac{W}{Z} - Ay + Bx). \end{aligned} \tag{6}$$

We can write these equations in the form:

$$u = u_t + u_r \qquad v = v_t + v_r \tag{7}$$

where (u_t, v_t) denotes the translational component of the optical flow and (u_r, v_r) the rotational component:

$$\begin{aligned} u_t &= \frac{-U + xW}{Z} & u_r &= Axy - B(x^2+1) + Cy, \\ v_t &= \frac{-V + yW}{Z} & v_r &= A(y^2+1) - Bxy - Cx. \end{aligned} \tag{8}$$

So far we have considered a single point P. To define the optical flow globally we assume that P lies on a surface defined by a function $Z = Z(X,Y)$ which is positive for all values of X and Y. With any surface and any motion of a camera we can therefore associate a certain optical flow and we say that the surface and the motion *generate* this optical flow.

Optical flow, therefore, depends upon the six parameters of motion of the camera and upon the surface whose images are analyzed. Can all these unknowns be uniquely recaptured solely from optical flow? The answer is no. To see this, consider a surface S_2 which is a dilation by a factor k of a surface S_1. Further, let two motions denoted by M_1 and M_2 have the same rotational component and let their translational components be proportional to each other by the same factor k (we will say that M_1 and M_2 are *similar*). Then the optical flow generated by S_1 and M_1 is the same as the optical flow generated by S_2 and M_2. This follows directly from the definition of optical flow (8). It is still an open question whether there are any other pairs of distinct surfaces and motions which generate the same optical flow.

Determining the motion of a camera from optical flow is much easier if we are told that the motion is purely translational or purely rotational. In the next two sections we will deal with these two special cases. Then we shall analyze the case where no *a priori* assumptions about the motion of the camera are made.

3. Translational Case

In this section we discuss the case where the motion of the camera is assumed to be purely translational. As before, let $\vec{T} = (U, V, W)$ be the velocity of the camera. Then the following equations hold (see (8)):

$$u_t = \frac{-U + xW}{Z} \qquad v_t = \frac{-V + yW}{Z}. \tag{9}$$

3.1. Similar Surfaces and Similar Motions

It will be shown next that if two flows generate the same optical flow, and we know that the motions are purely translational, then the two surfaces are similar and the two camera motions are similar. Let Z_1 and Z_2 be two surfaces and let $\vec{T}_1 = (U_1, V_1, W_1)^T$ and $\vec{T}_2 = (U_2, V_2, W_2)^T$ define two different motions of a camera, such that Z_1 and $\vec{T}_1$ and Z_2 and $\vec{T}_2$ generate the same optical flow:

$$u = \frac{-U_1 + xW_1}{Z_1} \qquad v = \frac{-V_1 + yW_1}{Z_1}, \tag{10}$$

$$u = \frac{-U_2 + xW_2}{Z_2} \qquad v = \frac{-V_2 + yW_2}{Z_2}. \tag{11}$$

Eliminating Z_1, Z_2, u and v from these equations we obtain:

$$\frac{-U_1 + xW_1}{-V_1 + yW_1} = \frac{-U_2 + xW_2}{-V_2 + yW_2}. \tag{12}$$

We can rewrite this equation as:

$$(-U_1 + xW_1)(-V_2 + yW_2) = (-U_2 + xW_2)(-V_1 + yW_1), \tag{13}$$

or:

$$U_1V_2 - xV_2W_1 - yU_1W_2 + xyW_1W_2 = U_2V_1 - xV_1W_2 - yU_2W_1 + xyW_2W_1. \tag{14}$$

Since we assumed that Z_1 and $\vec{T}_1$ and Z_2 and $\vec{T}_2$ generate the same optical flow, the above equation must hold for *all* x and y. Therefore the following equations have to hold:

$$\begin{aligned} U_1V_2 &= U_2V_1 \\ -V_2W_1 &= -V_1W_2 \\ -U_1W_2 &= -U_2W_1. \end{aligned} \tag{15}$$

These equations can be rewritten as:

$$U_1{:}V_1{:}W_1 = U_2{:}V_2{:}W_2 \tag{16}$$

from which it follows that Z_2 is a dilation of Z_1. It is clear that the scaling factor between Z_1 and Z_2 (or equivalently between $\vec{T}_1$ and $\vec{T}_2$) cannot be recovered from the optical flow, regardless of the number of points at which the flow is known. By *uniquely* determining the motion of the camera, we will mean that the motion is uniquely determined up to a constant scaling factor.

3.2. Least-Squares Formulation

In general, the direction of the optical flow at two points in the image plane determine the motion of a camera in pure translation uniquely. There is a drawback however to utilizing so little of the available information. The optical flow we measure is corrupted by noise and it is desirable to develop a robust method which takes this into account. Thus we suggest using a least-squares method [4], [14] to determine the movement parameters and the surface (i.e., the best fit with respect to some norm).

For the following we assume that the image plane is the rectangle $x\epsilon[-w, w]$ and $y\epsilon[-h, h]$. The same method applies if the image has some other shape. (In fact, it can be used on sub-images corresponding to individual objects in the case that the environment contains objects which may move relative to one another). Furthermore we have to assume that $1/Z$ is a bounded function and that the set of points where $1/Z$ is discontinuous is of measure zero. This condition on $1/Z$ assures us that all necessary integrations can be carried out. We wish to minimize the following expression:

$$\int_{-h}^{h}\int_{-w}^{w} [(u - \frac{-U + xW}{Z})^2 + (v - \frac{-V + yW}{Z})^2]\, dx\, dy. \tag{17}$$

In this case then, we determine the best fit with respect to the ML_2 norm which is defined as:

$$\| f(x,y) \| = \int_{-h}^{h}\int_{-w}^{w} [f(x,y)]^2\, dx\, dy. \tag{18}$$

The steps in the least-squares method are as follows: First we determine that Z which minimizes the integrand of (17) at every point (x, y). Then we determine the values of U, V and W which minimize the integral (17).

Let us introduce the following abbreviations:

$$\alpha = -U + xW \qquad \beta = -V + yW. \tag{19}$$

Note that the *expected* flow, given U, V and W is simply:

$$\bar{u} = \frac{\alpha}{Z} \qquad \bar{v} = \frac{\beta}{Z}. \tag{20}$$

Then we can rewrite (17) as:

$$\int_{-h}^{h}\int_{-w}^{w} [(u - \frac{\alpha}{Z})^2 + (v - \frac{\beta}{Z})^2]\, dx\, dy. \tag{21}$$

We proceed now with the first step of our minimization method. Differentiating the integrand of (17) with respect to Z and setting the resulting expression equal to zero yields:

$$(u - \frac{\alpha}{Z})\frac{\alpha}{Z^2} + (v - \frac{\beta}{Z})\frac{\beta}{Z^2} = 0. \tag{22}$$

Therefore we can write Z as:

$$Z = \frac{\alpha^2 + \beta^2}{u\alpha + v\beta}. \tag{23}$$

This equation, by the way, imposes a constraint on U, V and W, since Z must be positive. We do not make use of this except to help us pick amongst two opposite solutions for the translational velocity later on. Note that now:

$$u - \frac{\alpha}{Z} = \beta\frac{u\beta - v\alpha}{\alpha^2 + \beta^2} \qquad v - \frac{\beta}{Z} = -\alpha\frac{u\beta - v\alpha}{\alpha^2 + \beta^2} \tag{24}$$

and we can therefore rewrite (17) as:

$$\int_{-h}^{h}\int_{-w}^{w} \frac{(u\beta - v\alpha)^2}{\alpha^2 + \beta^2}\, dx\, dy. \tag{25}$$

It should be clear, by the way, that uniformly scaling U, V and W does not change the value of (25). This is a reflection of the fact that we can determine the motion parameters only up to a constant factor.

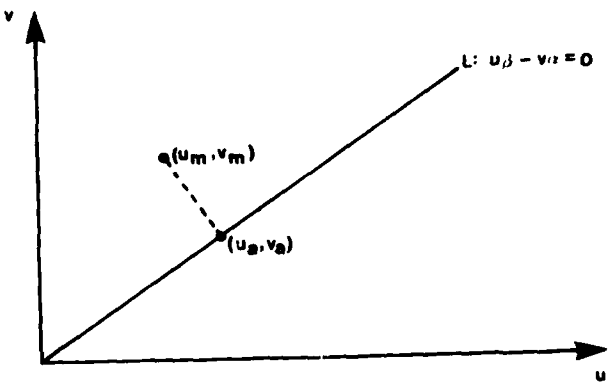


Figure 2. Geometrical Interpretation

Before proceeding with the second step, we give a geometrical interpretation in Figure 2 of what we have so far. Suppose that the motion parameters U, V, and W are given. At any given point, say (x_0, y_0), optical flow depends not only upon the motion parameters but also upon the value of Z at that point, Z_0 say. However, the direction of (u, v) does not depend upon Z_0. The point (u, v) must lie along the line L in the uv-plane defined by the equation $u\beta - v\alpha = 0$. Let the *measured* optical flow at (x_0, y_0) be denoted by (u_m, v_m), and let the closest point on the line L be (u_a, v_a). This corresponds to a particular Z_a given by (23). The remaining error is the distance between the point (u_m, v_m) and the line L. The square of this distance is given by the integrand of (25).

For the second step, we differentiate (25) with respect to U, V and W and set the resulting expressions equal to zero:

$$\int_{-h}^{h}\int_{-w}^{w} \frac{\beta(u\beta - v\alpha)(u\alpha + v\beta)}{(\alpha^2 + \beta^2)^2}\, dx\, dy = 0$$
$$-\int_{-h}^{h}\int_{-w}^{w} \frac{\alpha(u\beta - v\alpha)(u\alpha + v\beta)}{(\alpha^2 + \beta^2)^2}\, dx\, dy = 0 \tag{26}$$
$$\int_{-h}^{h}\int_{-w}^{w} \frac{(y\alpha - x\beta)(u\beta - v\alpha)(u\alpha + v\beta)}{(\alpha^2 + \beta^2)^2}\, dx\, dy = 0.$$

Let us introduce the following abbreviation:

$$K = \frac{(u\beta - v\alpha)(u\alpha + v\beta)}{(\alpha^2 + \beta^2)^2}. \tag{27}$$

Then equations (26) can be rewritten as:

$$\int_{-h}^{h}\int_{-w}^{w} [(-V + yW)K]\, dx\, dy = 0$$
$$-\int_{-h}^{h}\int_{-w}^{w} [(-U + xW)K]\, dx\, dy = 0 \tag{28}$$

$$\int_{-h}^{h}\int_{-w}^{w}[(-yU+xV)K]\,dx\,dy=0.$$

The sum of U times the first integral, V times the second integral, and W times the third integral is identically zero. Thus the three equations are linearly dependent. This is to be expected, for if:

$$f(kU,kV,kW)=f(U,V,W), \tag{29}$$

where f is a differentiable function and k a constant, then:

$$U\frac{\partial f}{\partial U}+V\frac{\partial f}{\partial V}+W\frac{\partial f}{\partial W}=0. \tag{30}$$

The result is also consistent with the fact that only two equations are needed, since the translational velocity can be determined only up to a constant factor. Unfortunately equations (28) are nonlinear in U,V and W and we are not able to show that they have a unique (up to a constant scaling factor) solution.

3.3. Using a Different Norm

There is a way, however, to devise a least-squares method which allows us to display a closed form solution for the motion parameters. Instead of minimizing (17), we will try to minimize the following expression:

$$\int_{-h}^{h}\int_{-w}^{w}[(u-\frac{-U+xW}{Z})^2+(v-\frac{-V+yW}{Z})^2] \times(\alpha^2+\beta^2)\,dx\,dy \tag{31}$$

obtained by multiplying the integrand of (17) by $\alpha^2+\beta^2$. Then we apply the same least-squares method as before to (31). When the measured optical flow is not corrupted by noise, both (31) and (17) can be made equal to zero by substituting the correct motion parameters. We thus obtain the same solution for the motion parameters whether we minimize (31) or (17). If the measured optical flow is not exact, then using expression (31) for our minimization, we *obtain the best fit with respect not to the ML_2 norm, but* to another norm which we call the $ML_{\alpha\beta}$ norm:

$$\| f(x,y)\|_{\alpha\beta}=\int_{-h}^{h}\int_{-w}^{w}[f(x,y)]^2(\alpha^2+\beta^2)\,dx\,dy. \tag{32}$$

What we have here is a minimization in which the error contributions are *weighted*, greater importance being given to points where the optical flow velocity is larger. This is most appropriate when the measurement of *larger* velocities is more accurate.

Which norm gives the best results depends on the properties of the noise in the measured optical flow. The first norm is better suited to the sitation where the noise in the measurements is *independent* of the magnitude of the optical flow. Note also that if we really want the minimum with respect to the ML_2 norm, we can use the results of the minimization with respect to the $ML_{\alpha\beta}$ norm as starting values in a numerical minimization.

We discuss now our least-squares method in the case where the norm is chosen to be $ML_{\alpha\beta}$. First we determine Z by differentiating the integrand of (31) with respect to Z and setting the result equal to zero. We again get (22):

$$(u-\frac{\alpha}{Z})\frac{\alpha}{Z^2}+(v-\frac{\beta}{Z})\frac{\beta}{Z^2}=0, \tag{33}$$

from which it follows that (23):

$$Z=\frac{\alpha^2+\beta^2}{u\alpha+v\beta}. \tag{34}$$

So we want to minimize:

$$\int_{-h}^{h}\int_{-w}^{w}(u\beta-v\alpha)^2\,dx\,dy. \tag{35}$$

Let us call this integral $g(U,V,W)$, then, since:

$$u\beta-v\alpha=(vU-uV)-(xv-yu)W, \tag{36}$$

we have:

$$g(U,V,W) = aU^2+bV^2+cW^2+2dUV+2eVW+2fWU, \tag{37}$$

where:

$$\begin{aligned} a&=\int_{-h}^{h}\int_{-w}^{w}v^2\,dx\,dy\\ b&=\int_{-h}^{h}\int_{-w}^{w}u^2\,dx\,dy\\ c&=\int_{-h}^{h}\int_{-w}^{w}(xv-yu)^2\,dx\,dy\\ d&=-\int_{-h}^{h}\int_{-w}^{w}uv\,dx\,dy\\ e&=\int_{-h}^{h}\int_{-w}^{w}u(xv-yu)\,dx\,dy\\ f&=-\int_{-h}^{h}\int_{-w}^{w}v(xv-yu)\,dx\,dy. \end{aligned} \tag{38}$$

Now $g(U,V,W)$ cannot be negative, and $g(U,V,W)=0$ for $U=V=W=0$. Thus a minimum can be found by inspection, but is not what we might have hoped for. In fact, to determine the translational velocity using our least-squares method we have to solve the following homogeneous equation for $\hat{T}$:

$$G\hat{T}=0 \tag{39}$$

where G is the matrix:

$$G=\begin{pmatrix} a & d & f\\ d & b & e\\ f & e & c\end{pmatrix}. \tag{40}$$

Clearly (39) has a solution other then zero if and only if the determinant of G is zero. Then the three equations (39) are linearly dependent and $\hat{T}$ can be determined up to a constant factor. In general, however, as the data is corrupted by noise, g cannot be made equal to zero for non-zero translational velocity and so $\vec{T} = (0,0,0)^T$ will be the only solution to (39). To see this in another way, note that g has the following form:

$$g(kU, kV, kW) = k^2 g(U, V, W) \tag{41}$$

where k is a constant. Clearly $g(U, V, W)$ assumes its minimal value for $U = V = W = 0$.

What we are really interested in, is determining the *direction* of $\vec{T}$ which minimizes g, for a fixed length of $\vec{T}$. Hence we impose the constraint that $\vec{T}$ be a unit vector. If $\vec{T}$ is constrained to have unit magnitude, the minimum value of g is the smallest eigenvalue of the matrix G and the value of $\vec{T}$ for which g assumes its minimum can be found by determining the eigenvector corresponding to this eigenvalue [8]. This follows from the observation that g is a quadratic form which can be written as:

$$g(U, V, W) = \vec{T}^T G \vec{T}. \tag{42}$$

Note that G is a positive semidefinite hermitian matrix as $a \geq 0$, $b \geq 0$, $c \geq 0$, $ab \geq d^2$, $bc \geq e^2$ and $ca \geq f^2$. (The last three inequalities follow from the Cauchy-Schwarz inequality [7], [8]). Hence all eigenvalues are real and non-negative and are the solutions λ of the third degree polynomial:

$$\begin{aligned} &\lambda^3 \\ &- (a + b + c)\lambda^2 \\ &+ (ab + bc + ca - d^2 - e^2 - f^2)\lambda \\ &+ (ae^2 + bf^2 + cd^2 - abc - 2def) = 0. \end{aligned} \tag{43}$$

There is an explicit formula for the least positive root in terms of the real and imaginary parts of the roots of the quadratic resolvent of the cubic. In our case this gives us the desired smallest root, since the roots cannot be negative. For the sake of completeness, however, various pathological cases that might come up will be discussed next, even though they are of little practical interest.

Note that $\lambda = 0$ is an eigenvalue if and only if G is singular, that is, if the constant term in the polynomial (43) equals zero. In fact, if the determinant of G is zero one can find a velocity $\vec{T}$ which makes g zero. It follows from a theorem in calculus that this happens only when the optical flow is either not corrupted by noise at all or only at a few points. The theorem states that if the integral of the square of a bounded and continuous function is zero then the function itself is zero. Hence errors can only occur at points where the optical flow is discontinuous, and these are exactly the points where the surface defined by Z is discontinuous. (These are also the places where existing methods for computing the optical flow [5] are subject to large errors).

It is impossible for exactly two eigenvalues to be zero, since this would imply that the coefficient of λ in the polynomial (43) equalled zero, while the coefficient of λ^2 did not. That in turn would imply that $ab = d^2$, $bc = e^2$, and $ca = f^2$, while a, b, and c are not all zero. For equality to hold in the Cauchy-Schwarz inequalities, however, u and v must both be proportional to $xv - yu$. This can only be true (for all x and y in the image) if $u=v=0$. But then all six integrals become zero and consequently all *three* eigenvalues are zero. This situation is of little interest, since it occurs only when the optical flow data is zero everywhere. Then the velocity is zero too. Once the smallest eigenvalue is known, it is straightforward to find the translational velocity which best matches the given data. To determine the eigenvector corresponding to an eigenvalue, say λ_1, we have to solve the following set of linear equations:

$$\begin{aligned} (a - \lambda_1)U + dV + fW &= 0 \\ dU + (b - \lambda_1)V + eW &= 0 \\ fU + eV + (c - \lambda_1)W &= 0. \end{aligned} \tag{44}$$

As λ_1 is an eigenvalue, equations (44) are linearly dependent. Let us for a moment assume that all eigenvalues are distinct, that is, the rank of the matrix $(G - \lambda I)$, where I is the identity matrix, is two. Then we can use any pair of them to solve for U, V in terms of W say. There are three ways of doing this. For numerical accuracy we may add the three results to get the symmetrical forms:

$$\begin{aligned} U &= (b - \lambda_1)(c - \lambda_1) - f(b - \lambda_1) \\ &\qquad - d(c - \lambda_1) + e(f + d - e) \\ V &= (c - \lambda_1)(a - \lambda_1) - d(c - \lambda_1) \\ &\qquad - e(a - \lambda_1) + f(d + e - f) \\ W &= (a - \lambda_1)(b - \lambda_1) - e(a - \lambda_1) \\ &\qquad - f(b - \lambda_1) + d(e + f - d). \end{aligned} \tag{45}$$

Note that λ_1 will be very small, if the data is good, and one may wish to just approximate the exact solution by using the above equations with λ_1 set to zero. (Then there is no need to find the eigenvalue). In any case, the resulting velocity may now be normalized so that its magnitude equals one. There is one remaining difficulty, arising from the fact that if $\vec{T}$ is a solution to our minimization problem, so is $-\vec{T}$. Only one of these solution will correspond to positive values of Z in equation (34) however. This can be easily seen by evaluating (34) at some point in the image. The case where the two smallest eigenvalues are the same will be discussed in one of the next paragraphs.

There is a simple geometrical interpretation of what we have done so far. To this end we consider the surface defined by $g(U, V, W) = k$ where k is a constant. Note that we can always find a new coordinate system $\tilde{U}, \tilde{V}, \tilde{W}$ in which $g(U, V, W)$ can be written as:

$$\lambda_1 \tilde{U}^2 + \lambda_2 \tilde{V}^2 + \lambda_3 \tilde{W}^2 = k \tag{46}$$

where λ_i for $i = 1, 2, 3$ are the three eigenvalues of the quadratic form. If the eigenvalues are all non-zero, the surface $g(U,V,W)=k$ is an ellipsoid with three orthogonal semi-axes of length $\sqrt{k/\lambda_i}$. We are particularly interested in the case where the constant k is the smallest eigenvalue. Then all three semi-axes have lengths less than or equal to one. Hence the ellipsoid lies within the unit sphere. If the two smallest eigenvalues are distinct, the unit sphere touches the ellipsoid in two places, corresponding to the largest axis. If the two smaller eigenvalues happen to be the same, however, the unit sphere touches the ellipsoid along a circle and as a result all the velocity vectors lying in a plane spanned by two eigenvectors give equally low errors. Finally, if all three eigenvalues are equal, no direction for $\vec{T}$ is preferred, since the ellipsoid becomes the unit sphere.

The case where exactly one eigenvalue is zero also has a simple geometrical interpretation. The surface defined by $g(U,V,W)=0$ is a straight line, which can be seen easily from the following equation:

$$\lambda_1 \tilde{U}^2 + \lambda_2 \tilde{V}^2 = 0 \tag{47}$$

written for the case when λ_3 is zero. (Remember that λ_1 and λ_2 are both positive.) Clearly the unit sphere intersects this line in exactly two points, one of them corresponding to positive values for Z in equation (34).

The method which we just described can be easily implemented. To this end, the problem can be discretized. An expression similar to (31) can be derived where the integrals are approximated by sums. Our minimization method can then be applied to these sums. The resulting equations are similar to ones described in this section, with summation replacing integration. We implemented the resulting algorithm and tested it using synthetic data including additive noise. The results agreed with our expectations.

One can use the ratio of the biggest to the smallest eigenvalue, the so called *condition number* [15], as a measure of confidence in the computed velocity. The result is very sensitive to errors in the measurements unless this ratio is much bigger than one.

Curiously, the same error integral as (35) is obtained in the case where the $ML_{Z_{uv}}$ norm is used:

$$\| f(x,y) \|_{Z_{uv}} \int_{-h}^{h}\int_{-w}^{w} [f(x,y)Z(x,y)]^2(u^2+v^2)\,dx\,dy. \tag{48}$$

We can arrive at a similar solution by multiplying the integrand in (17) by Z^2 instead of by $\alpha^2+\beta^2$. In that case the minimization is carried out with respect to the ML_Z norm defined by:

$$\| f(x,y) \|_Z = \int_{-h}^{h}\int_{-w}^{w} [f(x,y)Z(x,y)]^2\,dx\,dy. \tag{49}$$

Here optical flow velocities for points which are further away are weighted more heavily. This is most appropriate when the measurement of larger velocities is less accurate.

We end up with a quadratic form similar to g, only the integrals for the six constants corresponding to a, b, c, d, e, and f are a bit more complicated. Curiously they only depend on the direction of the optical flow at each point, not its magnitude.

Also, other constraints could be used. If we insist on $U^2+V^2=1$, for example, we obtain a quadratic instead of a cubic equation, and if we use $W=1$, a linear equation only need to be solved. The disadvantage of these approaches is that the result is sensitive to the orientation of the coordinate axes. Clearly, in the case of exact data, we get the right solution using any of the constraints mentioned above.

3.4. Using a Different Constraint

The minimization scheme discussed in the previous section gives us a unique solution in most cases for the velocity vector $\vec{T}$. Here we propose a slightly different approach which always gives us a unique solution. Note that applying the first step in our minimization method gives us a constraint between the values of Z, the velocity vector and the optical flow at every point. We can in addition assume that $Z=Z_0$ is known at a particular point, say (x_0,y_0). Using the $ML_{Z_{uv}}$ norm in our scheme, we want to minimize:

$$\int_{-h}^{h}\int_{-w}^{w} [uZ-(-U+xW)]^2+[vZ-(-V+yW)]^2 \times (u^2+v^2)\,dx\,dy. \tag{50}$$

Differentiating (50) with respect to Z, and setting the resulting expression equal to zero, we obtain:

$$Z = \frac{u\alpha+v\beta}{u^2+v^2}. \tag{51}$$

Thus we propose to solve the minimization scheme under the following constraint:

$$Z_0(u_0^2+v_0^2)-(u_0\alpha_0+v_0\beta_0)=0 \tag{52}$$

where u_0 and v_0 denote the components of the optical flow measured at (x_0,y_0) and α_0 and β_0 denote α and β evaluated at (x_0,y_0). The error integral (50) becomes after substituting (51):

$$\int_{-h}^{h}\int_{-w}^{w} (u\beta-v\alpha)^2\,dx\,dy \tag{53}$$

which is the same as (35) and is denoted by $g(U,V,W)$ (37). Thus we want to minimize:

$$g(U,V,W)+2\lambda[Z_0(u_0^2+v_0^2)-(u_0\alpha_0+v_0\beta_0)] \tag{54}$$

where λ denotes a Lagrangian multiplier. To determine U,V and W the following linear equations obtained by differentiating (54) with respect to U, V, W and λ have to be solved:

$$\begin{aligned} aU + dV + fW + \lambda u_0 &= 0 \\ dU + bV + eW + \lambda v_0 &= 0 \\ fU + eV + cW - \lambda(x_0u_0 + y_0v_0) &= 0 \\ u_0U + v_0V - (x_0u_0 + y_0v_0)W &= -Z_0(u_0^2 + v_0^2). \end{aligned} \tag{55}$$

These equations can be written in the form:

$$G_\lambda \vec{T}_\lambda = \vec{F} \tag{56}$$

where $\vec{T}_\lambda = (U, V, W, \lambda)^T$ and $\vec{F} = (0, 0, 0, -Z_0(u_0^2 + v_0^2))^T$. Let the determinant of G_λ be Δ_0:

$$\begin{aligned} \Delta_0 = \quad & (d^2 - ab)(x_0u_0 + y_0v_0)^2 \\ & +(e^2 - bc)u_0^2 + (f^2 - ac)v_0^2 \\ & +2[(de - bf)u_0(x_0u_0 + y_0v_0) \\ & +(df - ae)v_0(x_0u_0 + y_0v_0) \\ & +(cd - ef)u_0v_0]. \end{aligned} \tag{57}$$

Assuming that $\Delta_0 \neq 0$ we can easily determine $\vec{T}_\lambda$ from (55):

$$\vec{T}_\lambda = G_\lambda^{-1}\vec{F}. \tag{58}$$

Introducing the following abbreviation:

$$K = \frac{Z_0(u_0^2 + v_0^2)}{\Delta_0}, \tag{59}$$

we can give these formulae for $\vec{T}_\lambda$:

$$\begin{aligned} U &= K[u_0(bc - e^2) + v_0(ef - cd) \\ &\qquad + (x_0u_0 + y_0v_0)(bf - de)] \\ V &= K[u_0(ef - cd) + v_0(ac - f^2) \\ &\qquad + (x_0u_0 + y_0v_0)(ae - df)] \\ W &= K[u_0(de - bf) + v_0(df - ae) \\ &\qquad + (x_0u_0 + y_0v_0)(d^2 - ab)] \\ \lambda &= K[ae^2 + cd^2 + bf^2 - abc - 2def]. \end{aligned} \tag{60}$$

The disadvantage of this approach is that the result depends upon the values of the optical flow at a single point. To circumvent this problem we propose to determine average values for U, V and W in the following manner. First note that we are only interested in the ratios of U/W and V/W which obviously do not depend upon the (unknown) value for Z_0. Equivalently we could determine the value for K from the condition that $\vec{T}$ should have unit length. Hence we can determine values for U, V and W which depend only upon the values of the optical flow at a single point and the coefficient in the matrix G. If we want to remove the dependence of the result on the data at a single point, we can simply average the values obtained for all image points.

In the case where the data is exact, i.e., where the determinant of G, denoted by $\det G$, vanishes, the solution for $\vec{T}$ is the same one as obtained using no constraint in our minimizations scheme. To see this just observe that in that case $\lambda = 0$ as $\lambda = -K \det G$). In the case where $\Delta_0 = 0$, equations (55) have a solution only when $\det G = 0$. We do not have to be concerned with the case where $\Delta_0 = 0$ but $\det G \neq 0$ as we can argue that equations (55) always have to have a solution. Note that our method is based on the condition that Z is a certain function (51) of U, V, W. Hence (52) cannot impose a constraint which would be impossible to satisfy.

The methods discussed in this section have been applied to noisy synthetic data with the expected results.

4. Rotational Case

Suppose now that the motion of the camera is purely rotational. In order to determine the motion from optical flow we again use a least-squares algorithm with the ML_2 norm described in the previous section. Recall that in this case the optical flow is (see (8)):

$$\begin{aligned} u_r &= Axy - B(x^2 + 1) + Cy \\ v_r &= A(y^2 + 1) - Bxy - Cx. \end{aligned} \tag{61}$$

We will show now in an analogous fashion to section 3.1 that two different rotations, say $\vec{\omega}_1 = (A_1, B_1, C_1)^T$ and $\vec{\omega}_2 = (A_2, B_2, C_2)^T$, cannot generate the same optical flow. Assuming the converse, the following equations have to hold for all values of x and y:

$$\begin{aligned} & A_1xy - B_1(x^2 + 1) + C_1y \\ & \qquad = A_2xy - B_2(x^2 + 1) + C_2y \\ & A_1(y^2 + 1) - B_1xy - C_1x \\ & \qquad = A_2(y^2 + 1) - B_2xy - C_2x \end{aligned} \tag{62}$$

from which we can immediately deduce that $\vec{\omega}_1 = \vec{\omega}_2$.

In general, the direction of the optical flow at two points and its magnitude at one point determine the motion of a camera in pure rotation uniquely. We choose instead to minimize the following expression:

$$\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)^2 + (v - v_r)^2]\, dx\, dy. \tag{63}$$

As the motion is purely rotational, the optical flow does not depend upon the distance to the surface and therefore we may omit the first step in our method. Thus we immediately differentiate (63) with respect to A, B and C and set the resulting expressions equal to zero:

$$\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)xy + (v - v_r)(y^2 + 1)]\, dx\, dy = 0$$

$$\int_{-h}^{h}\int_{-w}^{w}[(u-u_r)(x^2+1)+(v-v_r)xy]\,dx\,dy=0 \quad (64)$$
$$\int_{-h}^{h}\int_{-w}^{w}[(u-u_r)y-(v-v_r)x]\,dx\,dy=0.$$

Rewriting the above equations we obtain:

$$\begin{aligned}\int_{-h}^{h}\int_{-w}^{w}&[uxy+v(y^2+1)]\,dx\,dy\\ &=\int_{-h}^{h}\int_{-w}^{w}[u_rxy+v_r(y^2+1)]\,dx\,dy\\ \int_{-h}^{h}\int_{-w}^{w}&[u(x^2+1)+vxy]\,dx\,dy\\ &=\int_{-h}^{h}\int_{-w}^{w}[u_r(x^2+1)+v_rxy]\,dx\,dy\\ \int_{-h}^{h}\int_{-w}^{w}&[uy-vx]\,dx\,dy\\ &=\int_{-h}^{h}\int_{-w}^{w}[u_ry-v_rx]\,dx\,dy\end{aligned} \quad (65)$$

and expanding these equations yields:

$$\begin{aligned}\bar{a}A+\bar{d}B+\bar{f}C&=\bar{k}\\ \bar{d}A+\bar{b}B+\bar{e}C&=\bar{l}\\ \bar{f}A+\bar{e}B+\bar{c}C&=\bar{m},\end{aligned} \quad (66)$$

where:

$$\begin{aligned}\bar{a}&=\int_{-h}^{h}\int_{-w}^{w}[x^2y^2+(y^2+1)^2]\,dx\,dy\\ \bar{b}&=\int_{-h}^{h}\int_{-w}^{w}[(x^2+1)^2+x^2y^2]\,dx\,dy\\ \bar{c}&=\int_{-h}^{h}\int_{-w}^{w}[x^2+y^2]\,dx\,dy\\ \bar{d}&=-\int_{-h}^{h}\int_{-w}^{w}[xy(x^2+y^2+2)]\,dx\,dy\\ \bar{e}&=-\int_{-h}^{h}\int_{-w}^{w}y\,dx\,dy\\ \bar{f}&=-\int_{-h}^{h}\int_{-w}^{w}x\,dx\,dy,\end{aligned} \quad (67)$$

and:

$$\begin{aligned}\bar{k}&=\int_{-h}^{h}\int_{-w}^{w}[uxy+v(y^2+1)]\,dx\,dy\\ \bar{l}&=-\int_{-h}^{h}\int_{-w}^{w}[u(x^2+1)+vxy]\,dx\,dy\\ \bar{m}&=\int_{-h}^{h}\int_{-w}^{w}[uy-vx]\,dx\,dy.\end{aligned} \quad (68)$$

If we call the coefficient matrix in (66) M and the column vector on the right-hand side $\vec{n}$, then we have:

$$M\vec{\omega}=\vec{n}. \quad (69)$$

Thus, provided the matrix M is non-singular, we can compute the rotation as follows:

$$\vec{\omega}=M^{-1}\vec{n}. \quad (70)$$

It is easy to see that the matrix M is non-singular in the special case of a rectangular image plane since then:

$$\begin{aligned}\bar{a}&=4wh(\frac{h^4}{5}+\frac{2h^2}{3}+1)+\frac{4w^3h^3}{9}\\ \bar{b}&=4wh(\frac{w^4}{5}+\frac{2w^2}{3}+1)+\frac{4w^3h^3}{9}\\ \bar{c}&=\frac{4wh}{3}(w^2+h^2)\\ \bar{d}&=\bar{e}=\bar{f}=0.\end{aligned} \quad (71)$$

So in the case of a rectangular image plane, the matrix is diagonal, which makes it particularly easy to compute its inverse. In fact, the matrix is diagonal if the image plane is symmetrical about the x-axis and the y-axis. As the extent of the image plane decreases, however, the matrix M becomes ill conditioned. That is inaccuracies in the three integrals ($\bar{k}$, $\bar{l}$, and $\bar{m}$) computed from the observed flow are greatly magnified. This makes sense since we cannot expect to accurately determine the component of rotation about the optical axis when observations are confined to a small cone of directions about the optical axis.

Again, in our numerical implementation of the algorithm the integrals in (67) can be approximated by sums. The methods discussed in this section have been applied to noisy synthetic data with the expected results.

5. General Motion

We would like now to apply a least-squares algorithm to determine the motion of a camera from optical flow given no *a priori* assumptions about the motion. It is plain that a least-squares method is easiest to use when the resulting equations are *linear* in all the motion parameters. Unfortunately, there exists no norm which will allow us to achieve this goal. We did find a norm, however, which resulted in equations that are linear in *some* of the unknowns and quadratic in the others. We again attack the minimization problem using the $ML_{\alpha\beta}$ norm under the constraint that $U^2+V^2+W^2=1$. The ensueing equations are polynomials in the unknowns U,V,W,A,B and C and can be solved by a standard iteration method like Newton's method or Bairstow's method [14] or by an interpolation scheme like *regula falsi* [14]. The expression we wish to minimize is:

$$\int_{-h}^{h}\int_{-w}^{w}\{[u-(\frac{\alpha}{Z}+u_r)]^2+[v-(\frac{\beta}{Z}+v_r)]^2\}(\alpha^2+\beta^2)\,dx\,dy. \quad (72)$$

The first step is to differentiate the integrand of (72) with respect to Z and set the resulting expression equal to zero:

$$Z = \frac{\alpha^2 + \beta^2}{(u - u_r)\alpha + (v - v_r)\beta}. \qquad (73)$$

We introduce the Langrangian multiplier λ as before and attempt to minimize:

$$\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)\beta - (v - v_r)\alpha]^2 \, dx\, dy + \lambda(U^2 + V^2 + W^2 - 1). \qquad (74)$$

The equations we have to solve to determine the motion parameters are obtained by differentiation:

$$\begin{aligned}
&\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)\beta - (v - v_r)\alpha] \\
&\qquad \times [-xy\beta + (y^2 + 1)\alpha]\, dx\, dy = 0 \\
&\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)\beta - (v - v_r)\alpha] \\
&\qquad \times [(x^2 + 1)\beta - xy\alpha]\, dx\, dy \quad = 0 \\
&\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)\beta - (v - v_r)\alpha] \\
&\qquad \times [y\beta + x\alpha]\, dx\, dy \quad = 0 \\
&\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)\beta - (v - v_r)\alpha] \\
&\qquad \times (v - v_r)\, dx\, dy + \lambda U \quad = 0 \\
&\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)\beta - (v - v_r)\alpha] \\
&\qquad \times (u - u_r)\, dx\, dy - \lambda V \quad = 0 \\
&\int_{-h}^{h}\int_{-w}^{w} [(u - u_r)\beta - (v - v_r)\alpha] \\
&\times [(u - u_r)y + (v - v_r)x]\, dx\, dy + \lambda W \quad = 0 \\
&U^2 + V^2 + W^2 \quad = 1.
\end{aligned} \qquad (75)$$

Note that the first three of these equations are linear in A, B and C from which these parameters can be determined uniquely in terms of U, V and W. Then we can determine U, V and W from the last four equations by a numerical method. To this end, the problem can be discretized and equations analogous to (75) derived, where summation of the appropriate expressions is used instead of integration.

6. Summary

Our objective was to devise a method for determining the motion of a camera from optical flow which allows for noise in the measured data. The least-squares method which we proposed in this paper meets our goal and is also suitable for numerical implementation. An important application of our results is in passive navigation. Here the path and instantaneous altitude of a vehicle is to be determined from information gleaned about the environment without the emission of sampling radiation from the vehicle.

Acknowledgements: The first author wants to thank Stephen Trilling for helpful comments which improved the style of this paper. We also want to thank Judi Jones for making the drawings.

7. References

[1] Bruss A.R., "Counterexample," unpublished manuscript, (1981).

[2] Fennema C.I. and Thompson W.B., "Velocity Determination in Scenes Containing Several Moving Objects," *Computer Graphics and Image Processing* Vol. **9**, No. 4, (1979), 301-315.

[3] Hadani I., Ishai G. and Gur M., "Visual Stability and Space Perception in Monocular Vision: Mathematical Model," *Journal of the Optical Society of America* Vol. **70**, No. 1, (1980), 60-65.

[4] Hildebrand F.B., *Introduction to Numerical Analysis*, McGraw-Hill, (1974).

[5] Horn B.K.P. and Schunk B.G., "Determining Optical Flow," *Artificial Intelligence* Vol. **17**, No. 1-3, (1981), 185-203.

[6] Koendrink J.J. and van Doorn A.J., "Local Structure of Movement Parallax of the Plane," *J. Opt. Soc. Am.* Vol. **66**, No. 7, (1976), 717-723.

[7] Kolmogorov A.N. and Fomin S.V., *Introductory Real Analysis*, Prentice-Hall Inc., (1970).

[8] Korn G.A. and Korn T.M., *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill Book Co., (1968).

[9] Longuet-Higgins H.C. and Prazdny K., "The Interpretation of a Moving Retinal Image," *Proc. Roy. Soc.*, Vol. **B**, No. 208, London, (1980), 358-397.

[10] Longuet-Higgins H.C., "A Computer Algorithm for Reconstructing a Scene from Two Projections," *Nature* Vol. **293**, (1981), 133-125.

[11] Meiri A.Z., "On Monocular Perception of 3-D Moving Object," *IEEE Transactions on Pattern Analysis and Machine Intel.igence* Vol. **PAMI-2**, No. 6, November (1980), 582-583.

[12] Nagel H.-H., "On the Derivation of 3D Rigid Point Configurations from Image Sequences," *IEEE Conference on Pattern Recognition and Image Processing* (1981).

[13] Prazdny K., "Determining the Instananeous Direction of Motion from Optical Flow Generated by a Curvilinear Moving Observer," *Computer Graphics and Image Processing* Vol. **17**, (1981), 238-248.

[14] Stoer J. and Bulirsch R., *Introduction to Numerical Analysis,* Springer-Verlag, (1980).

[15] Strang G., *Linear Algebra and its Applications,* Academic Press, (1976).

[16] Tsai R.Y. and Huang T.S., "Uniqueness and Estimation of 3-D Motion Parameters of Rigid Objects with Curved Surfaces," *Proc. of the IEEE Conference on Pattern Recognition and Image Processing, Las Vegas* (1981).

[17] Ullman S., *The Interpretation of Visual Motion,* The MIT Press, (1979).

AD-P000 127

A SYSTEM FOR AUTOMATED STEREO MAPPING

H. HARLYN BAKER and THOMAS O. BINFORD
Artificial Intelligence Laboratory
Computer Science Department
Stanford University
Stanford, California, 94305

Abstract

The past few years have seen a growing interest in the application of three-dimensional image processing. With the increasing demand for 3-D spatial information for tasks of passive navigation [Gennery 1980], [Moravec 1980], automatic surveillance [Henderson 1979], aerial cartography [Kelly 1977], [Panton 1978], and inspection in industrial automation, the importance of effective stereo analysis has been made quite clear. A particular challenge is to provide reliable and accurate depth data for input to object or terrain modelling systems (such as [Brooks 1981b]). This paper describes an algorithm for such stereo sensing. It uses an edge-based line-by-line stereo correspondence scheme, and appears to be fast, robust, and parallel implementable. The processing consists of extracting edge descriptions for a stereo pair of images, linking these edges to their nearest neighbors to obtain the edge connectivity structure, correlating the edge descriptions on the basis of local edge properties, then cooperatively removing those edge correspondences determined to be in error — those which violate the connectivity structure of the two images. A further correspondence process, using a technique similar to that used for the edges, is applied to the image intensity values over intervals defined by the edge correspondence. The result of the processing is a full image array disparity map of the scene viewed.

Area-based versus Edge-based Analysis

The development of commercial stereo mapping systems has been driven by requirements of terrestrial cartography. Cartography traditionally concentrated on mapping of terrain, producing elevation contour maps and digital terrain data bases. The bulk of stereophotogrammetry is accomplished manually by direct human compilation. There is growing use of recently developed interactive systems with automated stereo correlation capabilities — although these systems require extensive operator intervention (see [Friedman 1980]). All such systems use *area-based* cross-correlation. To the extent that cross-correlation is limited, they all have similar limitations. These limitations are in mapping accuracy and applicability to various terrain types (see [Ryan 1980] for a discussion of the errors and difficulties inherent in window shaping and cross-correlation).

Area-based stereo matching uses windowing mechanisms to isolate parts of two images for cross-correlation. *Edge-based* stereo matching uses two dimensional convolution operators to reduce an image to a depiction of its intensity boundaries, which can then be put into correspondence. Area-based cross-correlation techniques require distinctive texture within the area of correlation for successful operation. They break track:

- where there are ambiguous textures or featureless areas (roofs, sand and concrete);
- where the correlation area crosses surface discontinuities (at occlusions such as buildings, or thin objects (poles));
- where depth is ill-defined (such as through trees).

In general, these systems break track where there is no local correlation (zero signal, or where two images do not correspond) or where the correlation is ambiguous (where the signal is repetitive). The systems must be started manually and corrected when they break track.

Edge-based analysis appears to provide a solution to many of the problems of correlation. The system described here deals with edges, in its initial analysis, because of the:

a) *reduced combinatorics* — there are fewer edges than pixels,

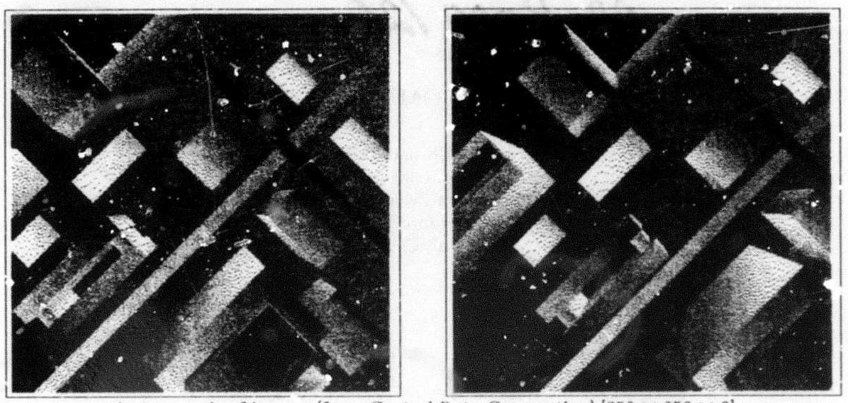
b) *greater accuracy* — edges can be positioned to sub-pixel precision, while area positioning precision is inversely proportional to window size, and considerably poorer, and

c) *more realistic invariance assumptions* — area-based analysis presupposes that the *photometric* properties of a scene are invariant to viewing position, while edge-based analysis works with the assumption that it is the *geometric* properties that are invariant to viewing position.

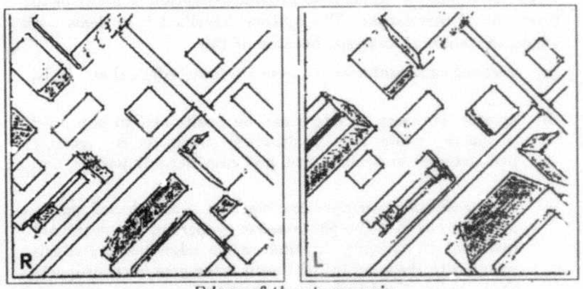
Method and Constraints

In the system to be described, edges are found in images by a simple convolution operator. They are located at positions in the *image where a change in sign of second difference in intensity* occurs. A particular operator, 1 by 7 pixels in size[1], measures the directional first difference in intensity at each pixel. Second differences are computed from these, and changes in sign of these second differences are used to interpolate zero crossings (*i.e.* peaks in first difference). Certain local properties other than *position* are measured and associated with each edge — *contrast*, *orientation*, and *intensity to either side* — and *links* are kept to nearest neighbours above, below, and to the sides. It is these properties that define an edge and provide the basis for the correspondence process (see the discussions of edge matching in [Arnold 1980], [Baker 1980]).

[1] The edge operator is simple, basically one dimensional, and is noteworthy only in that it is fast and fairly effective.


A stereo pair of images (from Control Data Corporation) [256 × 256 × 6]
Figure 1

The correspondence is a search for *edge* matches across images. Figure 2 shows the edges found in the two images of Figure 1 with the second difference operator (**note**: all stereo pairs in this paper are drawn for cross-eyed viewing). Although the operator works in both horizontal and vertical directions, it only allows matching on edges whose horizontal gradient lies above the noise — one standard deviation of the first difference in intensity. With no prior knowledge of the viewing situation, one could have any edge in one image matching any edge in the other. The depictions of Figure 2 have about 3500 edges each — the combinatorics of a naive matching strategy here would clearly be enormous.



Edges of the stereo pair
Figure 2

By constraining the geometry of the cameras during picture taking one can vastly limit the computation that is required in determining corresponding edges in the two images. If two equalivalent cameras are arranged with axes parallel, as shown in Figure 3, then they can be conceived of as sharing a single common image plane. Any point in the scene will project to two points on that joint image plane (one through each of the two lens centers), the connection of which will produce a line parallel to the baseline between the cameras. Thus corresponding edges in the two images must lie along the same line in the joint image plane. This line is termed an ***epipolar line*** (see [Hallert 1960]). If the baseline between the two cameras happens to be parallel to the scanning axis of the cameras, then the correspondence only need consider edges lying along matched lines parallel to that axis in the two images. These lines are termed ***conjugate***. Figure 3 indicates this camera geometry — a geometry which produces ***registered*** images. The algorithm described assumes the stereo pair to be registered, and if this is not the case then the appropriate transformation of one image relative to the other must be made before further processing is done. Note that a less restrictive solution would be to have the correspondence process informed of the camera geometries, and have it solve for the more general epipolar situation.

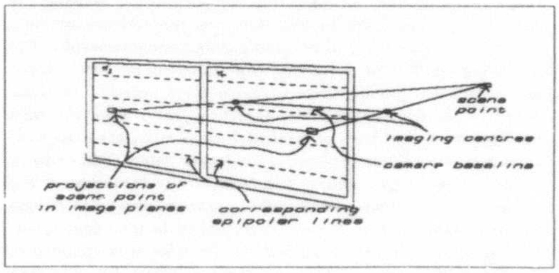
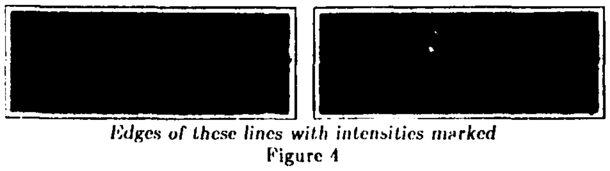
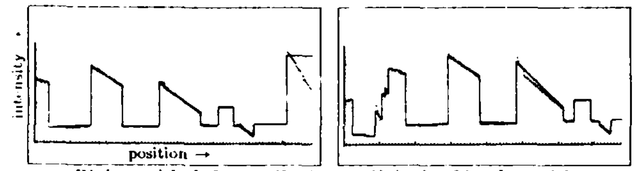

scene point
imaging centres
camera base line
projections of scene point in image planes
corresponding epipolar lines
Specialized Epipolar geometry
Figure 3

Figure 5 shows a pair of conjugate lines from the stereo pair of Figure 1, while Figure 4 plots the actual intensities of these lines, seen as dots, superimposed on the edges determined by the edge operator. Since one needs to compare edges only from conjugate lines in the two images, the matching process can be applied to the edges shown in Figure 4. The correspondence could proceed at this point by searching for the 'best' assignment of edges along such conjugate lines — an assignment which optimizes some goodness measure. However, normal combinatoric search is quite inadequate here. Typical lines have upwards of $n = 30$ edges each and the combinatoric space, with a naive upper bound of $n!$, grows rapidly with n (the CDC imagery above is not typical, rather it is synthetic, and actually quite noise-free — in contrast see the images of Figure 10). Even with extensive heuristic pruning, runs with a combinatoric search approach often take seconds per line ... and sometimes minutes (on a DEC KL-10). A superior approach to the correspondence task lies in using the Viterbi algorithm [Forney 1973], a dynamic programming technique used extensively in speech processing, and first used in vision research in some recent work at Control Data Corporation [Henderson 1979]. An earlier use of a similar dynamic programming technique for stereo matching is documented in [Gimel'farb 1972].

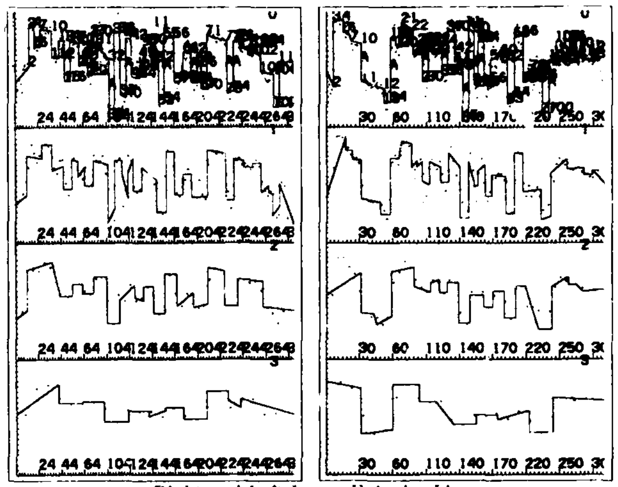

Edges of these lines with intensities marked
Figure 4



Right and Left Image Conjugate Epipolar Line Intensities
Figure 5

What distinguishes the Viterbi technique from normal search is the requirement that one be able to partition the original problem into two subproblems, each of which can be solved optimally and whose results can be processed to yield a global optimum for the original problem ('optimal' with respect to an evaluation function on the chosen parameters). In a recursive way, each of the subproblems may be divided and the solution process repeated. Geometrically, the partitioning constraint here is one of ***monotonicity*** of edge order. A left right ordering of edges in one image cannot correspond to a right left ordering in the other; *i.e.* there can be no positional reversals of edges in the image plane. This constraint allows one to make n tentative assignments of an edge on one line with the edges of the conjugate line in the other image, with each tentative assignment partitioning the correspondence problem into two subproblems. The two subproblems are the matching of edges lying to the **left** and **right** of the selected edge on the one line with edges lying to the **left** and **right** respectively of its tentative match on the other line. The optimality criterion selects the series of such assignments judged 'best'. This constraint excludes from analysis, for the time being, features such as ***wires or overhanging surfaces*** which lead to positional reversals in the image. This reversal also causes the human vision system trouble — we can fuse one or the other, the nearer or the farther, but not both at the same time (see [Burt 1980]).

The correspondence process could use the edges as were indicated in Figure 2 above, but in the interests of robustness and efficiency a different approach is taken here. A large amount of small scale detail in the images will increase the cost of the correlation exponentially. Reducing the level of detail and narrowing the extent of the required search will reduce the computation time and enhance noise immunity. This is achieved through the use of a ***coarse-to-fine*** analysis in which a reduced resolution matching process is first applied to bring the two images into rough correspondence. The removal of the small scale detail brings quite a reduction in the number of edges to be dealt with. Successive refinements in resolution bring successively finer detail into the analysis, and each such phase can use the results of the previous lower resolution analysis to narrow its search. Such an approach has had previous successful application in visual processing (*e.g.* [Marr 1977], [Grimson 1980], [Moravec 1980]), and has relevant ties to the neurophysiology of vision, where it is felt a multiple spatial frequency analysis is part of the human system's processing [Wilson 1978] (although the filtering used here is low pass, and not bandpass). It was our intent to use the low resolution components of the images to determine local approximate ***disparities***, and to use these as guides for the full resolution matching. To obtain the resolution reduction we use a linear smoothing filter to successively halve image resolutions, continuing until the image noise content reaches an acceptably low point — one brightness level (smoothing reduces noise, so increases signal-to-noise ratio). Figure 6 shows the edges in successive resolution reductions of a sample line pair from the images of Figure 10, again with dots marking the intensities.


Right and Left Image Epipolar Line Successive Resolution Reductions
Figure 6

The same basic second difference operator is used throughout the resolution reduction analyses, but its size and noise-based thresholds are altered to keep it matched to the characteristics of the 'new' reduced resolution image. These lowest resolution edges are matched in a manner to be described below. The ***intervals*** specified between nearest-corresponding edge pairs and their mates in the other image define local disparities to be used by the full resolution correspondence process.

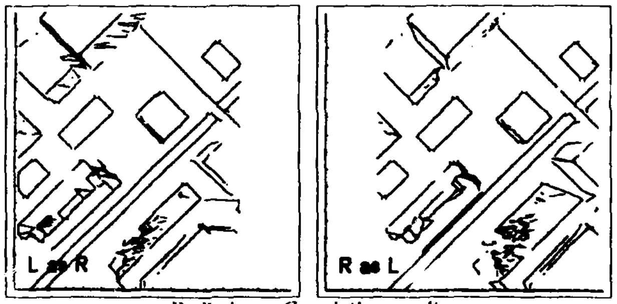
The Edge Matching Process

The process of determining edge correspondences is basically the same for both the reduced resolution and the full resolution processes: the only difference is in the set of parameters used by the optimization function. Full resolution matching uses ***edge orientation, side intensities, relative disparity*** (as measured by the reduced resolution phase), and ***interval compression*** implied by the correspondence (which uses edge position to determine the foreshortening of scene surfaces — analogous to human spatial frequency processing, as in [Blakemore 1970]). In reduced resolution matching ***side intensities, contrast,*** and ***interval compression*** are used. These parameter measures all enter the computation as probabilistic weightings: $0 \leq P \leq 1$.

Each edge from a line in the reference image is associated with a set of possible matching edges from the conjugate line in the other image (including the null match, which would imply that the edge is either spurious or is obscured in the other image). Slightly complicating this, each such edge is treated as a doublet, being a **left** side (the termination of the interval to its left) and a **right** side (the start of the interval to its right); left sides of edges can only match left sides of edges, right sides only right sides (quite obviously). This left-right distinction is essential in domains where surfaces may occlude one another, leaving a surface to one side of an edge hidden from one viewpoint while it is visible to the other. Each side of an edge is termed a *half-edge*. For each pairing in the set of possible matches the ***static*** probability of correspondence is determined — this is the product of all of the mentioned probability measures except ***interval compression*** (which is determined dynamically). The optimization process then uses these probabilities, composing them with the dynamic ***interval compression*** probabilities in determining the 'best' correspondence of half-edges along a line in one image with half-edges along the corresponding line in the other image (details in [Baker 1981]). This computation is $O(n^3)$ for n edges along either image line — it would be $O(n^2)$ were it not for the use of ***interval compression*** probabilities.

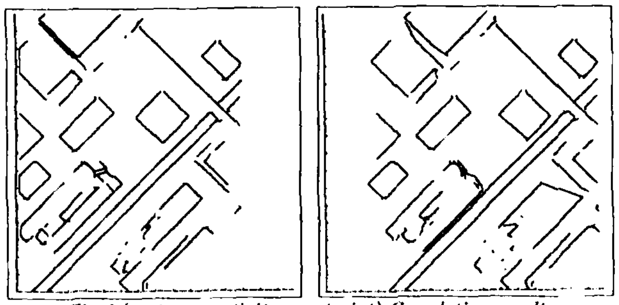
The Connectivity Constraint

Figure 7 shows the results of the whole image line-by-line correspondence process. Wherever there is a noticeable horizontal jag across the image, there is an error in the matching. What is really being depicted here is ***change in disparity*** along connected edges in each image. This is achieved by plotting between the connected edges of an image, but rather than using just each edge's coordinate, use its coordinate plus associated disparity. Thus, when a connected stretch of edges in one image is matched to various parts of the other image, the drawing will jump horizontally back and forth in the other image's space, touching the various parts matched with the connected stretch. At these horizontal jumps, the process is suggesting that there is a large change in depth. This is a suggestion of a ***break in depth continuity***

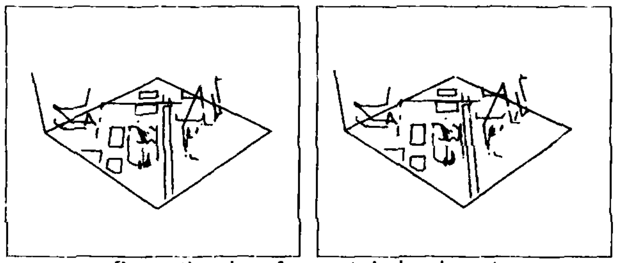


Preliminary Correlation results
Figure 7

Let us emphasize, the dynamic programming algorithm above performs a ***local*** optimization for the correlation of individual lines in the image — it uses no information outside of those lines. A very strong ***global*** constraint is apparent and available here, that of ***edge connectivity***. It may be presumed (by general position) that, in the absence of other information, a connected sequence of edges in one image should be seen as a connected sequence of edges in the other, and that the structure in the scene underlying these observations may be inferred to be a continuous surface detail or a continuous surface contour. A cooperative procedure uses this connectivity assumption to remove edge correspondences which violate surface continuity. The evidence for these mismatches is found through the tracking of disparities along connected edges on adjacent lines (this is what Figure 7 depicts). The results of the matching after this process has functioned are shown in Figure 8 (with the same type of depiction as Figure 7). Figure 9 shows a perspective view of these connected edge elements in depth. Notice that this figure drawn by the system shows that it has truly captured something of the 3-D structure of the scene.



Final (post-connectivity constraint) Correlation results
3000 half-edge correlate pairs
Figure 8

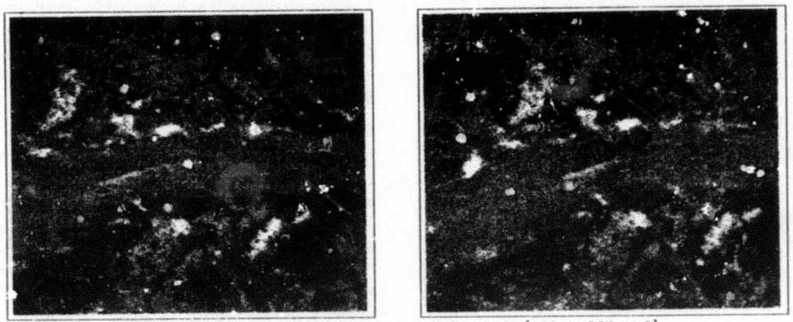


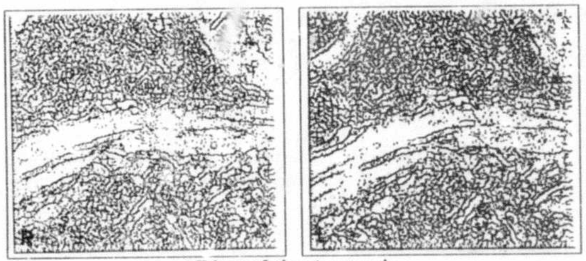
Perspective view of connected edge elements
(the z axis is disparity, not depth)
Figure 9

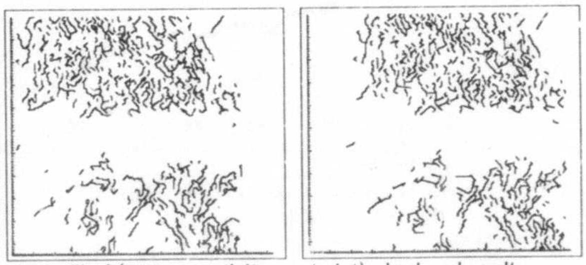
Figures 11 through 14 show the various stages of the edge-based stereo correspondence process when applied to the image pair of Figure 10. This data (provided by the Night Vision Laboratory of the U.S. Army), an aerial view of natural terrain, is considerably noisier and significantly more detailed than the synthetic urban imagery of Figure 1. It more clearly demonstrates the importance of the interline connectivity constraint and, as shown in Figure 6, the use of resolution reduction during the correspondence process.

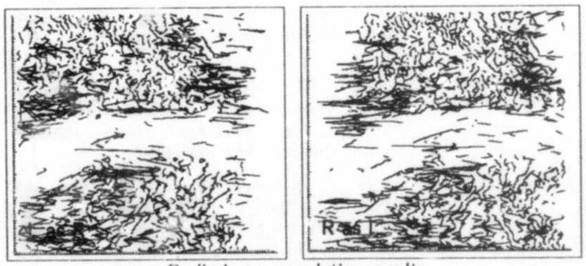
Intensity Based Matching

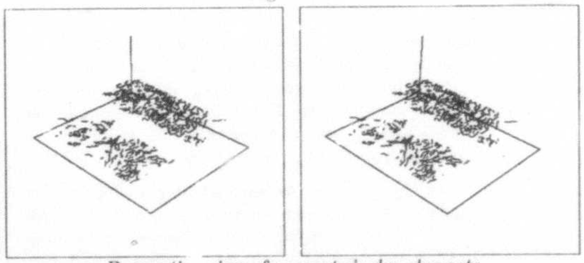
The description so far has been of an edge-based stereo correspondence scheme — one which uses a Viterbi optimality condition and a cooperative continuity enforcement process in establishing reliable matches between the intensity boundaries, or ***edges***, of a stereo pair of images. Yet there is much more information one could provide about the depth in scenes such as those depicted in Figures 1 and 10. The 3-D edge descriptions of Figures 9 and 14 just highlight the structure of these scenes, giving rather sparse disparity measures. One would like fuller stereo detail from the matching, and a subsequent correspondence process, this time based on image ***intensity*** values, supplies this.


NVL stereo pair of images — natural terrain [168 × 200 × 9]
Figure 10


Edges of the stereo pair
Figure 11

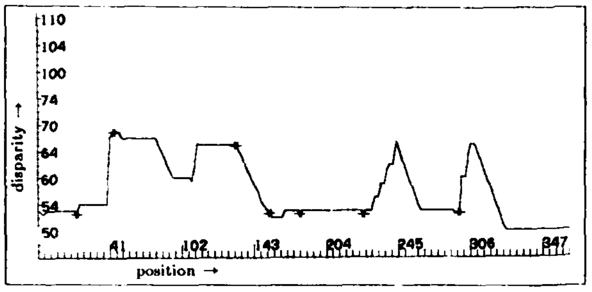

Final (post-connectivity constraint) edge-based results
3700 half-edge correlate pairs
Figure 13

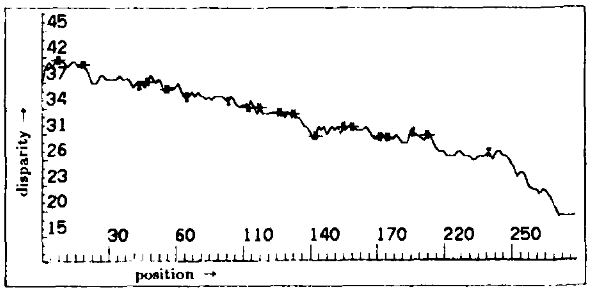

Preliminary correlation results
Figure 12


Perspective view of connected edge elements
Figure 14

The above ***edge-based*** matching, in indicating corresponding edges in the two images, provides strong local 'vergence' information which greatly constrains the matching problem for any remaining correspondence process. One can take unpaired ***edges*** from an interval along one epipolar line and match them (again, via Viterbi) with unpaired edges from the corresponding interval of the conjugate line in the other image (the intervals are bounded by either matched edge pairs or the periphery of the image). This matching serves to "fill in the gaps" of the primary edge-based correspondence. A final correspondence process (the fourth!) takes ***intensity*** values from the intervals between paired edges along conjugate lines and does yet one more Viterbi matching on them. We are still developing the metrics used in this correspondence process — at the present we use 1) intensity variance and 2) deviation from linearity of the interpolated surface. Figure 15 indicates the results of this processing on two sample lines (DC line above, NVL below). Arrowheads (→ and ←) show half-edge pairings (to the *left* side and to the ***right*** side, respectively), and the plotted contours show the disparity values assigned by the intensity correspondence process (depth can easily be determined from disparity once the camera parameters are known).

Figures 16 and 18 show stereo perspective views of the full correspondence results for the two sets of imagery (disparities are those of the left camera image), and Figures 17 and 19 show single views of the same surfaces at full resolution. These figures show that the correspondence algorithm produces a full image array disparity map of the viewed scenes. We have not yet measured its accuracy.





Epipolar Line disparities (←→ correlated edges)
Figure 15

Performance

The correlation algorithm described here provides this three-dimensional sensing in a *fast, robust,* and *parallel implementable* way.

[*fast*] The analyses as shown in Figures 9 and 14 took roughly 25 and 35 seconds apiece from image input to the final edge correspondence results as depicted. The remaining edge and intensity correspondences of Figures 17 and 19 required a further 45 and 15 seconds, respectively (on a KL-10).

[*robust*] The use of line-by-line matching, each processed independently (accumulating a good deal of redundant evidence), and the use of a coarse-to-fine strategy (where the more reliable lower frequency components are correlated first) have been seen to give a good basis for obtaining the correct global consensus in the subsequent cooperative process.

[*parallel implementable*] Since there is no interline dependence during the various correspondence processes, and the subsequent cooperative process has only pairwise interline interactions, there is a high potential for a parallel (*n*-processors for *n* lines) realization.

Ahead

There is still, of course, considerable research to be done within this depth determination process. We will be:

- looking into improvement issues in the present algorithm, such as having the matcher use colour information and using stronger global information to both aid stereopsis and eliminate false correspondences, and
- testing it on further stereo imagery, both aerial and near range, using digital terrain models for accuracy tests where they are available.

Further afield, we are interested in developing such a correspondence scheme into a continuous, multi-image correlator, capable of integrating analyses over a series of passively sensed stereo views in building a highly accurate and detailed map of scene depth.

Our group's plans in stereo research do not end with 3-D sensing. Stereo for us is a tool for obtaining better descriptions of the environment. These descriptions are to be used in interpreting scenes, ***for our goal is image understanding***. ACRONYM [Brooks 1981b] is a very successful geometric modelling and reasoning system developed here at Stanford over the past five years. It has the capability of manipulating three-dimensional models of structures that it has been taught and interpreting presented imagery in the context of these models. At present it is handicapped with strictly two-dimensional image data. To remove this restriction, the primary thrust of our stereo research over the next few years will be in developing a rule-based stereo mapping system to work within the ACRONYM system. A stereo mapping system as described here will be one of the components of this rule-based system.

Acknowledgements

This research has been supported by the Defense Advanced Research Projects Agency of the United States Government on contract MDA 903-80-C-0102.

References

[Arnold 1980]	Arnold, R.D., and T.O. Binford, "Geometric Constraints in Stereo Vision," *Soc. Photo-Optical Instr. Engineers,* vol. 238, Image Processing for Missile Guidance, 1980, 281-292.
[Baker 1980]	Baker, H. Harlyn, "Edge Based Stereo Correlation," *Proc. ARPA Image Understanding Workshop,* University of Maryland, April 1980, 168-175.
[Baker 1981]	Baker, H. Harlyn, "Depth from Edge and Intensity Based Stereo,"forthcoming AI memo, Stanford Artificial Intelligence Laboratory, Summer 1982.
[Blakemore 1970]	Blakemore, Colin, "A New Kind of Stereoscopic Vision," *Vision Research,* vol. 10, 1970, 1181-1199.
[Brooks 1981b]	Brooks, Rodney A., "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence Journal,* vol. 16, 1981.
[Burt 1980]	Burt, Peter and Bela Julesz, "A Disparity Gradient Limit for Binocular Fusion," *Science,* vol. 208, no. 9, May 1980, 615-617.
[Forney 1973]	Forney, G. [illegible] vid Jr., "The Viterbi Algorithm," *Proc. IEEE,* vol. 61, no. 3, March 1973, 268-278.
[Friedman 1980]	Friedman, S.J., editor, Manual of Photogrammetry, American Society of *Photogrammetry,* C. C. Slama, editor-in-chief, 1980.
[Gennery 1980]	Gennery, Donald B., "Modelling the Environment of an Exploring Vehicle by Means of Stereo Vision," Ph.D. thesis, Stanford Artificial Intelligence Laboratory, AIM-339, June 1980.

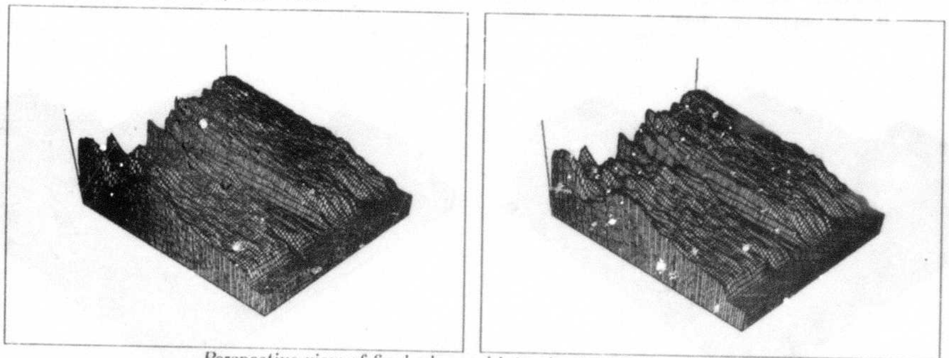
[Marr 1977] Marr, D. and T. Poggio, "A Theory of Human Stereo Vision," MIT Artificial Intelligence Memo no. 451, November 1977.

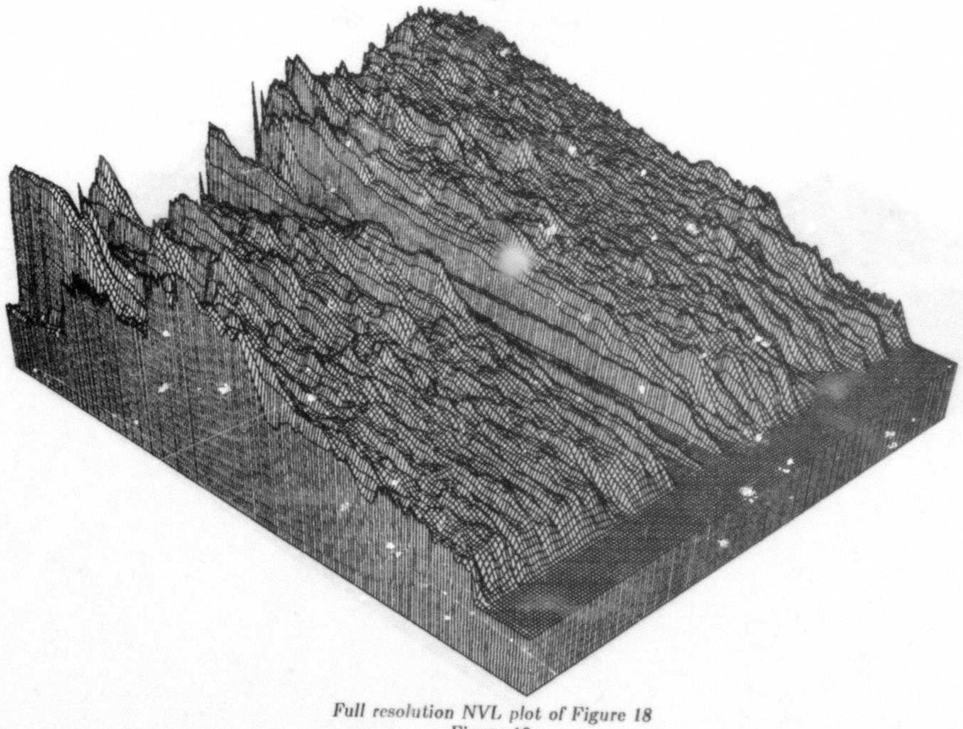
[Moravec 1980] Moravec, Hans P., "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Stanford Artificial Intelligence Laboratory, AIM 340, Ph.D. thesis, September 1980.

[Panton 1978] Panton, Dale J., "A Flexible Approach to Digital Stereo Mapping," *Photogrammetric Engineering and Remote Sensing*, vol. 44, no. 12, December 1978, 1499–1512.

[Ryan 1980] Ryan, Thomas W., and B.R. Hunt, "The Prediction of Accuracy in Digital Cross-Correlation of Stereo-Pair Images," *Soc. Photo-Optical Instr. Engineers*, vol. 219, Electro-Optical Technology for Autonomous Vehicles, 1980.

[Wilson 1978] Wilson, Hugh R., "Quantitative Prediction of Line Spread Function Measurements: Implications for Channel Bandwidths," *Vision Research*, vol. 18, 1978, 493–496.


Perspective view of final edge and intensity correlation — NVL
after median filtering
Figure 18


Full resolution NVL plot of Figure 18
Figure 19

[Gimel'farb 1972] Gimel'farb, G.L., V.B. Marchenko, and V.I. Rybak, "An Algorithm for Automatic Identification of Identical Sections on Stereopair Photographs," *Kybernetica* (translations) no. 2, March–April 1972, 311–322.

[Grimson 1980] Grimson, W.E.L., "Computing Shape Using a Theory of Human Stereo Vision," Department of Mathematics, MIT, June 1980.

[Hallert 1960] Hallert, Bertil, *"Photogrammetry, Basic Principles and General Survey,"* McGraw-Hill Book Company Inc., 1960.

[Henderson 1979] Henderson, Robert L., Walter J. Miller, C.B. Grosch, "Automatic Stereo Recognition of Man-Made Targets," *Society of Photo-Optical Instrumentation Engineers*, vol. 186, *Digital Processing of Aerial Images*, August 1979.

[Kelly 1977] Kelly, R.E., P.R.H. McConnell, and S.J. Mildenberger, "The Gestalt Photomapping System," *Journal of Photogrammetric Engineering and Remote Sensing*, vol. 43, 1407, 1977.

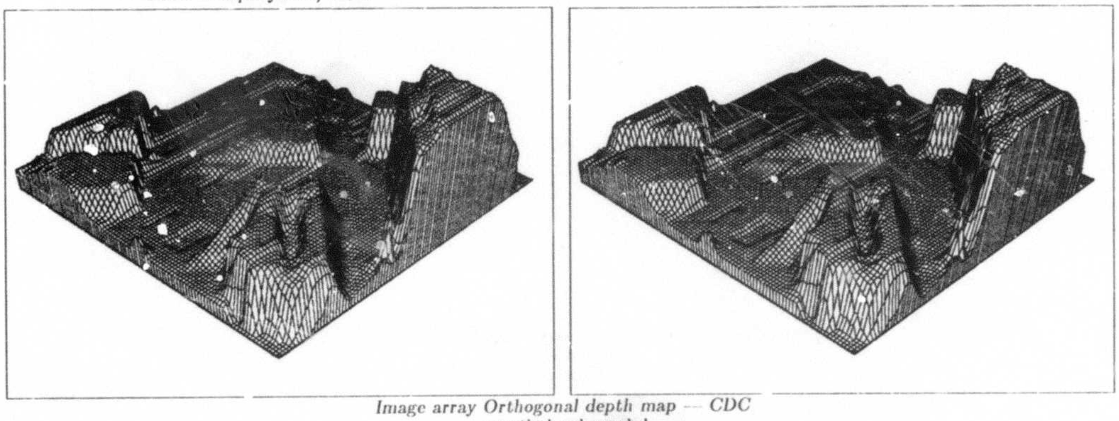
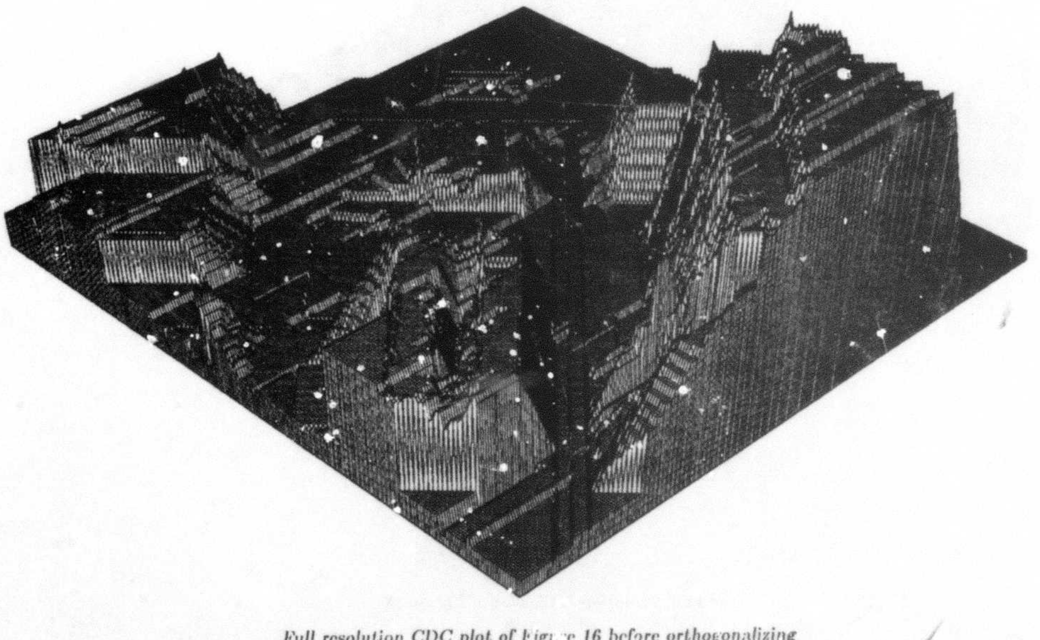


Image array Orthogonal depth map — CDC
smoothed and sampled
Figure 16


Full resolution CDC plot of Figure 16 before orthogonalizing
Figure 17

SECTION III

TECHNICAL PAPERS
(NOT PRESENTED)

AD-P000 128

SEGMENTATION IN ACRONYM

David H. Marimont

Artificial Intelligence Laboratory
Stanford University, Stanford, California, 94305

ABSTRACT

The role of segmentation in the ACRONYM vision system is to detect, group, and represent low-level image structures such as edges and curves without detailed knowledge of objects in the scene or of viewpoint. These structures are then passed to higher levels of processing which do use such knowledge to predict the appearance of objects in images and then to match the predictions with the observed structures. Currently this matching takes place at the level of ribbons, the perspective projections of the generalized cylinders from which ACRONYM's models are built. The three steps in finding ribbons are the detection and linking of edges, the segmentation of linked edges into curves, and the grouping of curves into ribbons; of these, the first two have recently been significantly improved, and revision of ribbon grouping operations is under way. Formerly, linked edges were segmented only into straight lines, so that providing curved data for grouping into ribbons should lead to superior performance.

1: INTRODUCTION

A limiting factor in the performance of model-based vision systems has long been the quality of the low-level features extracted from images. Without good low-level data, a system searching for occurences of models in images is unable to take advantage of the subtle cues to three dimensional interpretation on which human perception seems to depend so heavily; for a discussion of such cues see [Binford 1981]. The goal of the work described here is the accurate detection of the image features which underlie some of these cues: edges, smooth curves, corners in curves, and groups of related curves. Without accurate knowledge of the characteristics of and interrelationships among such features, inferences from cues expressed in terms of these features cannot be drawn with high reliability.

2: WHAT IS SEGMENTATION?

In what follows, the term segmentation refers to the process of building a data-driven symbolic description of an image without using detailed knowledge of specific objects or of viewpoint. In some cases, such information is available and should be used. In its absence, segmentation relies on generic knowledge and careful analysis of the signal. First, significant low-level structures in the image are detected, such as edges, curves, and 'well-formed' regions. Next these structures are grouped according to very general principles applicable in a broad range of circumstances; an abbreviated list includes proximity, collinearity, parallelism, and connectivity. Finally, suitable representations for these grouped structures are computed and passed to higher levels of processing. In model based vision, the goal is often to find instances of previously stored models in an image. To this end, higher levels use knowledge of specific objects to match the predicted appearance of objects in images with the grouped structures discovered during segmentation.

3: THE ROLE OF SEGMENTATION IN ACRONYM

In ACRONYM, the matching between grouped structures discovered during segmentation and the appearance of models predicted by higher levels of processing takes place at the ribbon level. A brief digression into ACRONYM's modeling system is necessary here. A ribbon is the perspective projection of a generalized cylinder; all models in ACRONYM are built from generalized cylinders, volumetric primitives first introduced in [Binford 1971]. A generalized cylinder is the volume swept out by a simple, closed plane curve as it moves along a space curve; the plane curve may be scaled as it moves.

In a somewhat oversimplified version of the prediction process, the starting point is a model consisting of a group of generalized cylinders, so the projection of the model is equivalent to that of the group of generalized cylinders. But since a ribbon is the perspective projection of a generalized cylinder, the perspective projection of the model is merely a group of ribbons. Thus higher levels of ACRONYM predict structures to be sought in the image in terms of ribbons. More precisely, higher levels of processing predict the appearance of a model by specifying the ribbons likely to be observed if the predicted object is present in the scene. For more details of prediction in ACRONYM, see [Brooks 1981].

The role of segmentation is to detect all the ribbons in the image and to pass them to this higher level of processing. The process by which ribbons are found has recently been improved and now consists of the following steps. First, points in the image which lie on edges are detected and linked into extended edges. Here an edge is defined as a discontinuity in image intensity. Next, corners in the extended edges are found and smooth curves are fitted to the extended edges between them. Finally, the curves are grouped into ribbons.

These operations are being implemented in ACRONYM in four processing stages. Initially, the image is laterally inhibited by convolving it with a mask designed to remove the effects of smooth shading, which can pose problems for edge detectors. In the subsequent edge detection and linking, points on edges are located to subpixel accuracy and linked into extended edges based on local information, so that both detection and linking are completed in the same pass over the image. Next, during curve segmentation, each extended edge is processed to find corners in the edge, to smooth the edge between the corners, and then to approximate the edge with a smooth curve. In the final, ribbon finding stage, curves are grouped into ribbons by such operations as jumping gaps in curves and matching extended smooth curves opposite one another to form two sides of a ribbon.

Currently, the first three of these stages are implemented in ACRONYM, leaving only the ribbon finder to be completed in improved form. Figures 1-4 contain examples of these processing steps for several typical images. Figure 1 is a bin of connecting rods, courtesy of GM; figure 2 is an aerial photograph of an airplane sitting on the runway at San Francisco International Airport; figure 3 is a PC board, courtesy of SRI; and figure 4 is an aerial photograph of a water treatment plant in Fort Belvoir, Virginia.

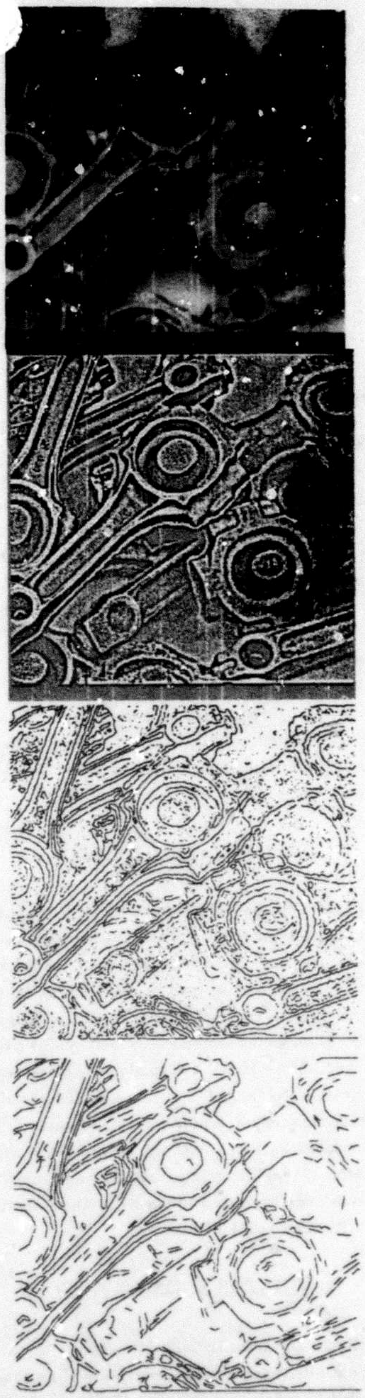
The previous segmentation algorithm consisted of the Nevatia-Babu line finder [Nevatia 1978] and a ribbon finder. This line finder incorporates edge detection, linking, and fitting of straight lines to linked edges; the ribbon finder then groups these straignt lines into ribbons. Because the new segmentation algorithm will find more edges, will find them more accurately, will find fewer edges caused by smooth shading, and will segment edges into curved elements, the completion of the new ribbon finder to group these curved elements into ribbons should significantly improve ACRONYM's performance.

4: DETAILS OF THE IMPLEMENTATION

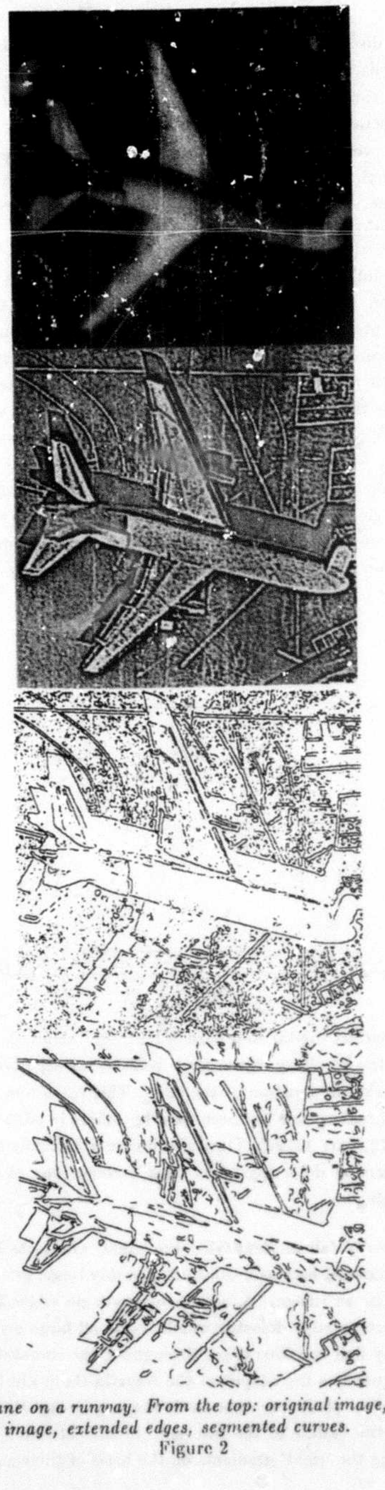
As noted above, the four steps in ACRONYM's segmentation algorithm are lateral inhibition, edge detection and linking, curve segmentation, and ribbon finding. What follows describes each of these steps in some detail.

4.1 Lateral inhibition

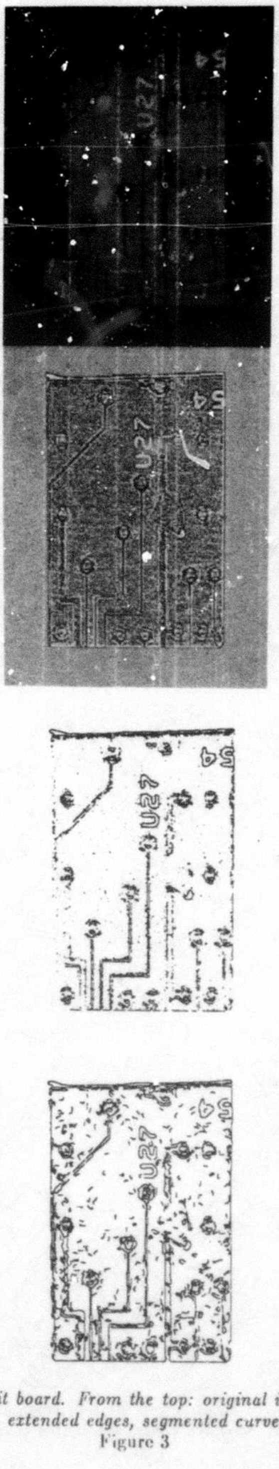
An edge is a discontinuity in the intensity of the continuous, unquantized image which must be sampled and quantized (i.e. digitized) before it can be conveniently manipulated by a digital computer. Unfortunately, sampling and quantization make impossible a generally meaningful definition of discontinuity.



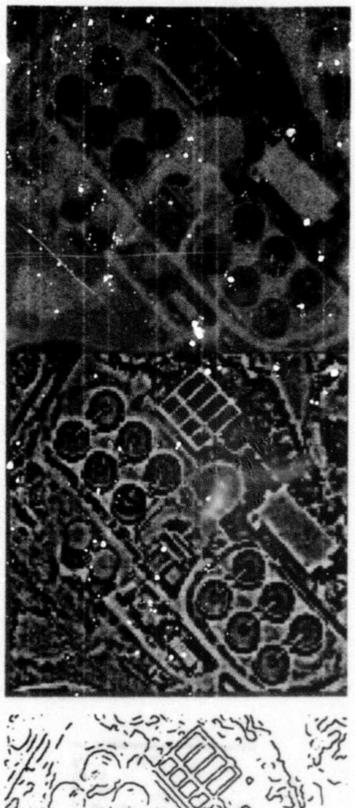
A bin of connecting rods. From the top: original image, laterally inhibited image, extended edges, segmented curves.
Figure 1



An airplane on a runway. From the top: original image, laterally inhibited image, extended edges, segmented curves.
Figure 2

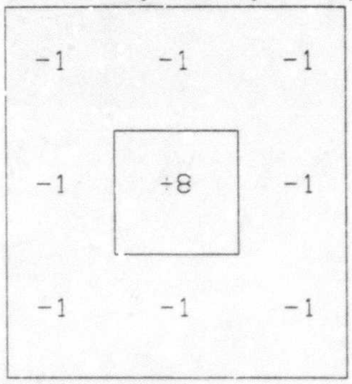


A printed circuit board. From the top: original image, laterally inhibited image, extended edges, segmented curves.
Figure 3


A water treatment plant. From the top: original image, laterally inhibited image, extended edges, segmented curves.
Figure 4

Instead, discontinuities in the continuous, unquantized image often appear merely as gradients in the digital image. The fact that the converse is not true is a major obstacle to accurate edge detection. That is, the problem is that not every gradient, not even every large gradient, corresponds to a discontinuity in the original, continuous, unquantized image. Thus there needs to be some way of distinguishing gradients which correspond to discontinuities from those which do not.

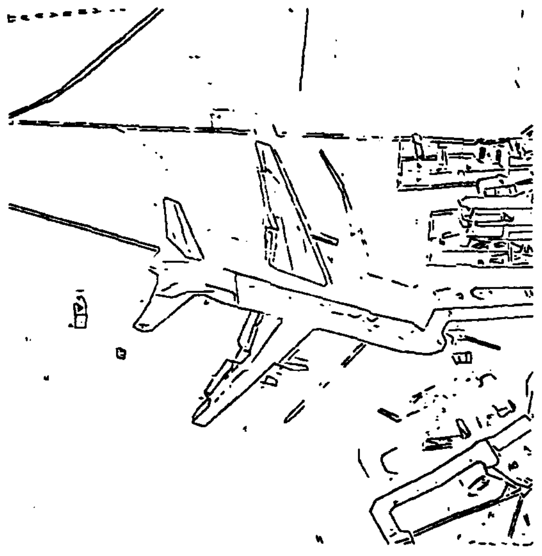
Laterally inhibiting the image before detecting edges helps avoid a common pitfall in distinguishing between these two types of gradients; it is discussed below. The operation itself is simply a convolution of the image with a rotationally symmetric mask with positive center, negative surround, and zero mean. Whenever the mask 'sits' on a portion of the image where the intensities increase linearly in any direction with any magnitude, the result is zero. The current implementation of ACRONYM uses a 'difference of boxes' mask, where the entire mask and its central region are square, as in figure 5. The rotational symmetry is quite rough, but keeping track of column sums during the convolution enables the computations to proceed very quickly.



"Difference of boxes" mask for lateral inhibition.
Figure 5

When a perfect digital step edge is laterally inhibited, the transition of the resulting signal from positive to negative, or vice versa, marks the location of the edge. This transition is known as a zero crossing and has been used by others in edge detection [Horn 1972,Marr 1979a]. Once the image is laterally inhibited, the problem of detecting edges thus becomes one of detecting zero crossings.

A common pitfall in identifying the 'right' gradients is smooth shading, defined here as a constant, possibly large, gradient over a region in an image. A human observes no edges in such a region, but an edge detector which treats all large gradients as edges may find spurious edges throughout the smoothly shaded area. Figure 6 is the output of the Nevatia Babu line finder for a wider view of the airplane in 2; shading causes the spurious edges in the middle of the fuselage. In this situation, at least, identifying the 'right' gradients on the basis of their magnitude alone is too simple a strategy. Lateral inhibition avoids this pitfall because it is equivalent to subtracting the linear portion of the signal from itself. Thus laterally inhibiting smooth shading, or a constant gradient, sends the signal to zero because it has only a linear component. Since no zero crossings result, no edges are detected, and the desired correspondence with human performance is achieved.



Result of the Nevatia-Babu line finder for a wider view of the airplane in figure 2.

Figure 6

The second row of figures 1, 2, 3, and 4 show the results of laterally inhibiting some typical images. The brighter regions correspond to positive values, the darker to negative, and a medium gray to zero; the zero-crossings therefore appear as the boundaries between light and dark regions.

4.2 Edge detection and linking

For the reasons just stated, zero crossings of the laterally inhibited signal are used to localize points on edges. Once a zero crossing between a pair of horizontally or vertically adjacent pixels is found, the subpixel location of the underlying edge is approximated by linear interpolation between the values of the laterally inhibited signals at the two pixels. The strength is approximated by the difference between the values and used later during the processing to remove noise.

The zero-crossings are linked into extended edges as they are detected. Each junction of four pixels is examined, once the zero-crossings between each horizontally and vertically adjacent pair of the four have been detected. If there are only two, they are linked together; if there are four, they are linked in pairs so that the direction of the transition from positive to negative is constant along each linked edge. (Naturally, if *only one* zero crossing is present, no linking is possible, and there cannot be three transitions from positive to negative without a fourth.)

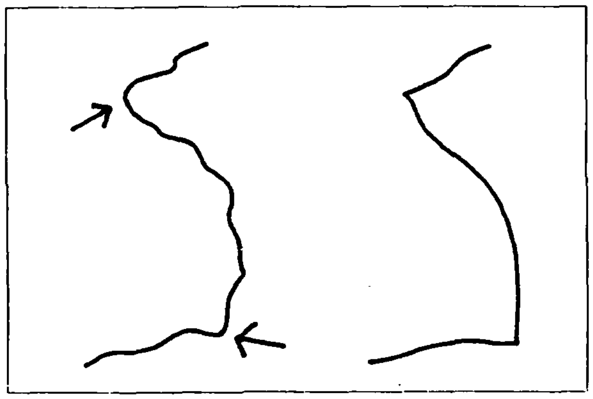
One traversal of the picture suffices for both detecting zero crossings *and linking them into extended edges*. The algorithm that makes this possible is roughly as follows. Consider visiting each pixel by sweeping each row from left to right, starting at the top row and working down, and suppose the next pixel to be visited is in the middle of the picture somewhere. A record has been kept of the extended edges linked to zero crossings between horizontally adjacent pixels in the row above, and of the extended edge linked to that between the vertically adjacent pair of pixels in the column to the left, of the same row and the row above. Now if there is a zero crossing between the current pixel and its neighbor on the left or that above, the existing extended edges to which it can be linked are available from these records, and the newly-detected zero crossing can easily be added to the appropriate extended edge. The records are updated if necessary so that yet-to-be-detected zero crossings can be added to the newly-extended edges when appropriate, and the detecting and linking continues. Thus keeping track of at most one row's plus one cell's worth of extended edges at a time enables the detection of zero crossings and their linkage into extended edges to be completed in one pass.

The third row of figures 1, 2, 3, and 4 contain sample output from the edge detection and linking phase for some typical images.

4.3 Curve segmentation

Each extended edge is segmented by examining it individually, locating corners, and fitting smooth curves to approximate the extended edge between the corners. The goal is to extract the important structural features of each extended edge which tend to be obscured by sensor noise, imperfections in the imaging system, digitization, and artifacts of previous stages of processing. For example, one might expect edges in the perfectly sensed, continuous image underlying the noisy, digital image to be characterized by smoothly varying tangents over most of their length punctuated by occasional discontinuities in tangent. The processing of extended edges seeks to reconstruct these two important features by locating the tangent discontinuities, less formally called corners, and by estimating the smoothly varying tangent.

Since the extended edges are digital curves, there is once again no generally useful definition of tangent discontinuity. Moreover, *corners in the extended edge are usually rounded or confounded* with small but sharp corners caused by noise. For these reasons, the criterion for detecting corners is based on extrema in a digital measure of the curvature of the extended edge. The extremum occurs at the largest change in tangent in regions where all changes in tangent may be large and thus handles blurred corners correctly. A threshold on the measure of curvature is used so that corners too small to be distinguished from random noise are ignored. The corner-finding process is for a single edge is illustrated by the left half of figure 7.



Left, corners detected in an extended edge; right, smooth curves fitted to edge between corners.
Figure 7

Once the corners in an extended edge have been located, it is assumed that the edge in the perfectly sensed, continuous image underlying the extended edge has a smoothly varying tangent between each pair of adjacent corners. Any small corners and wiggles in the extended edge between corners are attributed to noise and suppressed by the subsequent fitting of a smooth curve. Thus the smooth curve represents an estimate of the smoothly varying tangent of the underlying continuous edge. This estimate is constructed by smoothing samples of the tangent to the extended edge and fitting cubic splines to them. The right half of figure 7 contains an example of this step for a single edge.

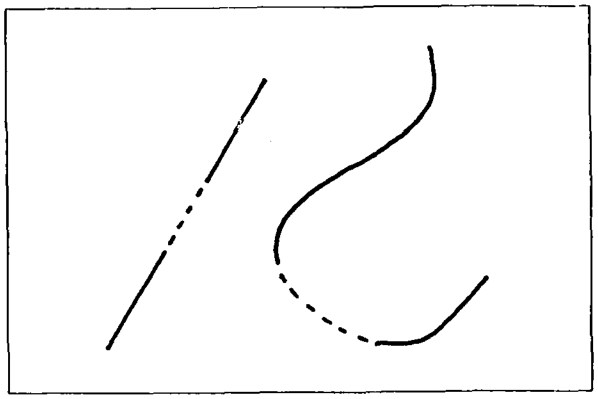
The current implementation of the curve segmenter incorporates many of these ideas. The remainder will soon be included. A current topic of research is the development of a spline based on estimates of curvature to replace the cubic spline now used, which is based on estimates of tangent.

The bottom row of figures 1, 2, 3, and 4 contain sample output from the curve segmenter for some typical images.

4.4 Ribbon finding

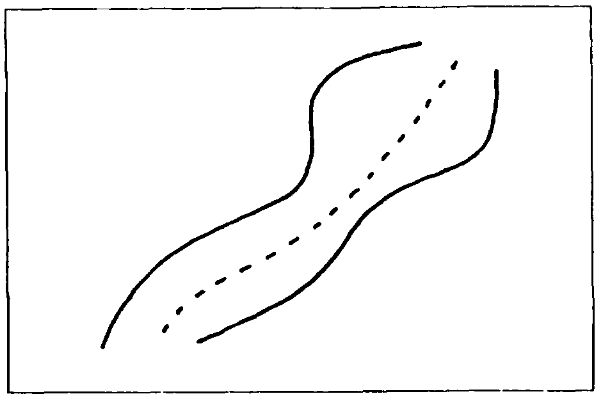
The ribbon finder tries to group the curves produced by the curve segmenter into ribbons. A variety of grouping operations are in the process of being implemented. Here only a few of the more important are mentioned to sketch in broad outline the style of computation involved in this stage of segmentation.

The linking of collinear lines is a simple but important operation. The higher order analogue is the jumping of gaps in curves that was mentioned earlier. Here, two separate curves that can be interpreted as one long, smooth curve with a single break are joined together. Recognizing when this is appropriate involves not just the positions and orientations of the two curves, but their curvatures as well. The efficient storage and retrieval of these data in a two-dimensional domain requires sophisticated algorithms and data structures whose exploration is an active topic of research. These operations are illustrated by figure 8.



Left, linking collinear lines; right, jumping gaps in curves.
Figure 8

A somewhat more complex operation is the matching of two extended, opposing curves to form the longer opposing boundaries of a ribbon. Figure 9 provides an example. One way to do this is point-to-point matching from one curve to another using only information local to the points being matched. Algorithms which perform a more global curve-to-curve correspondence based on representing curves in terms of their curvatures at several different resolutions are being developed.



Two extended curves with opposing boundaries grouped to form a ribbon; the dotted line indicates the ribbon's axis.
Figure 9

Again, the ribbon finder that uses curved data from the curve segmenter is not yet fully implemented, so no results are available. For examples of the performance of the old ribbon finder, which grouped only straight lines into ribbons, see [Brooks 1979].

6: ACKNOWLEDGEMENTS

The ACRONYM vision system was conceived primarily by Thomas Binford and Rodney Brooks and implemented primarily by Rodney Brooks. Michael Lowry implemented the Nevatia-Babu line finder and the fast lateral inhibition algorithm. Thomas Binford and David Lowe designed the curve segmenter; David Lowe was primarily responsible for implementing it. The work reported here was performed under ARPA contract N00039-82-C-0250.

7: REFERENCES

[Binford 1971] Binford, T.O., "Visual Perception by Computer," invited paper at the *IEEE Conference on Systems, Science and Cybernetics*, Miami, December 1971.

[Binford 1981] Binford, Thomas O., "Inferring Surfaces from Images," *Artificial Intelligence*, vol. 17(1981), August 1981, 205-244.

[Brooks 1979] Brooks, Rodney A., "Goal-directed Edge Linking and Ribbon Finding," *Proceedings of the ARPA Image Understanding Workshop*, Menlo Park, California, April 1979, 72-76.

[Brooks 1981] Brooks, Rodney A., "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence Journal*, vol. 16, 1981.

[Horn 1972] Horn, B.K.P., "The Binford-Horn Edge Finder," MIT AI Memo 285, 1972, revised December 1973.

[Marr 1979a] Marr, D. and E. Hildreth, "Theory of Edge Detection," MIT Artificial Intelligence Memo no. 518, April 1979.

[Nevatia 1978] Nevatia, R. and K.R. Babu, "Linear Feature Extraction," *Proc. ARPA Image Understanding Workshop*, Pittsburgh, November 1978, 73-78.

AD-P000 129

SCENE ANALYSIS USING REGION-BASED CONSTRAINT FILTERING

Les Kitchen

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

A general-purpose scene-analysis system is described which uses constraint-filtering techniques to apply domain knowledge in the interpretation of the regions extracted from a segmented image. An example is given of the configuration of the system for a particular domain, FLIR (Forward Looking InfraRed) images, as well as results of the system's performance on some typical images from this domain.

1. Introduction

An image (whether on the human retina, on photographic film, or in some electronic device) is formed by a complicated interaction of light with objects in three-dimensional space (a scene). Scene analysis is the process of unravelling this interaction: inferring from an image the arrangement of lighting and objects that produced it. In theory, this problem is indeterminate: A given image may result from many different scenes, all of which happen to appear identical from the observer's viewpoint. But in practice there are usually sufficient restrictions on allowable scenes to permit essentially only one interpretation of the image. The problem is to find this interpretation efficiently. Humans are clearly able to do this. Can computers achieve similar performance?

In this paper we present a method for scene analysis based on the application of constraint-filtering techniques to a network of regions extracted from an image. Such an approach has two chief advantages. First, its conceptual simplicity: It provides a clean separation between the general processing algorithm and the knowledge about a particular domain, which is expressed declaratively as constraints. Second, its computational speed: Constraint-filtering can be decomposed into many almost independent processes which can all be run in parallel on a suitable multiprocessor computer.

To try this approach, we have implemented a prototype system that does scene analysis by constraint filtering. A diagram giving an informal overview of the system is shown in Figure 1. In the interest of expediency we have made many simplifications. For example, only a few crude measurements are made on the extracted regions. In the following sections, we will describe the prototype system, taking note of these simplifications. We will also show how the system is used in a particular domain -- forward-looking infrared (FLIR) images of battlefield scenes. This domain was chosen partly because of its military interest, but primarily because its moderate complexity is just about right for fully exercising the prototype system. Then we will discuss the system's performance, taking care to distinguish those failures that are inherent in the method from those that are merely the result of simplifications made in this implementation, and finally, we will suggest directions for further progress.

2. Segmentation

A digital image is merely an array of light intensity (or color) values. There seems to be no way of going directly from these values to a description of a scene in terms of the objects in it. As argued by Barrow and Tenenbaum [1,2], Marr [3], and numerous others, several stages of processing are needed, each with its own intermediate representations of the information contained in the image. A first step is to organize the pixels into groupings that correspond more closely to the objects in the scene.

Typically, this is done by segmenting the image into regions of fairly homogeneous brightness. For many scenes, this is a reasonable thing to do. In most cases, the regions will correspond to the objects themselves, or else to significant pieces of them. By this means the myriads of pixels in an image can be reduced to a few score, or a few hundred regions, considerably decreasing the amount of data that must be processed, but with little loss of information. Furthermore, since regions more closely correspond to objects, expectations about the appearance of objects can be more readily applied to the regions than to unorganized pixels.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Clara Robertson in preparing this paper. The author would like to express his debt to Azriel Rosenfeld, Teresa Silberberg, Larry Davis, and Michael Shneier for helpful and stimulating discussions during the development of this system and to Robert Kirby for his quite excellent implementation of LISP.

However, segmentation into regions is not without problems. An initial process of segmentation normally applies a single criterion of homogeneity over the entire image. Unfortunately, a difference in brightness that is insignificant in some contexts (such as a fluctuation in a textured background) may well be very significant in other contexts (such as part of the border of an object with its surroundings). Most segmentation processes take little account of this sort of contextual information, and so make errors of two sorts: oversegmentation and undersegmentation. Oversegmentation breaks into pieces what should ideally be a single region. Properties of the region as a whole (such as shape and area) and relations with other regions (such as adjacency and surroundedness) cannot be computed directly, but can only be recovered by attempting to merge pieces that are likely to belong to the same region. More serious is undersegmentation, by which several regions that should be distinct are fused together. Again, region properties and relations are lost, but recovering them is a more difficult business of attempting to split the fused region into parts.

Several attempts have been made to overcome this problem. Tenenbaum and Barrow in IGS (Interpretation Guided Segmentation) [4] used domain knowledge to guide the low level segmentation. Constraints about the relationships between objects were used to guide the merging of pixels into regions. Feldman and Yakimovsky [5] also used semantic constraints to guide segmentation. Another approach, used by Nagao and Matsuyama [6], first performs as unguided segmentation and later corrects the errors in this segmentation by a semantically controlled process of merging and splitting regions. We assume that undersegmentation never occurs, and that oversegmentation is not serious: that an object is at worst broken into two or three pieces. We augment our domain model to cover fragments of objects, but without making any attempt to integrate them into wholes. For the simple domain used as an example, the initial segmentation can usually be fine-tuned by hand to fit our assumptions above. Even so, failures are not uncommon, indicating that a more subtle treatment of segmentation errors is needed.

First we smooth the image using an edge-preserving smoothing technique in order to reduce noise. The particular technique used does not matter greatly, but usually we have used Narayanan and Rosenfeld's histogram-guided smoothing technique [7], which has proved quite effective. Next, we requantize the image into a small number of gray levels (typically five), following the peak structure of the histogram of the smoothed image.

After this, the regions themselves can be extracted by a connected components analysis. At the same time, we make a few measurements on each region; these measurements serve as a description of the region for all subsequent processing. We construct the bounding upright rectangle around each region (see Figure 2) and measure the image location of its lower left corner, its width and height, as well as the area and average brightness of the region itself.

As mentioned above, these measurements provide only a crude description of each region, but sufficient for this prototype system. A full implementation would need a more complete description of shape, perhaps the chain code of the boundary of each region. Since any region description is necessarily incomplete, it may ultimately be necessary to refer back to the original image to check for properties that cannot easily or efficiently be extracted by preprocessing operations.

3. Constraint filtering

After segmentation, scene analysis becomes mostly a matter of labelling the regions with their identifications as objects or object parts. (For now we ignore the problem of organizing the parts of objects into wholes.) Clearly, only those labellings are valid that can be derived from an arrangement of real objects in space. Properties of objects, and relationships between them, imply corresponding properties and relationships of the image regions that result from these objects. These projected properties and relationships constrain the possible labelling of regions with object identifications. Thus scene analysis can be reduced to a constraint satisfaction problem. (The early work of Barrow et al. [8,9], used this approach, with the constraints derived from a relational structure which provided a single but inflexible scene model.)

The traditional technique for solving such problems is backtracking. However, backtracking is inherently a sequential technique, which does not lend itself well to parallel processing. Even if we restrict our attention to sequential processing, simple backtracking has a serious defect, especially on the problems arising in scene analysis: It suffers greatly from "thrashing" behavior [10, 11,12]. When a failure is discovered, only the most recent labelling is reconsidered. If the true cause of the failure lies in an earlier labelling, it will take the program many steps of blind backtracking before it can undo the incorrect labelling. To overcome these problems, a number of authors [10,11,14,15,16,17,18,21] have proposed "constraint filtering", "relational consistency", or "discrete relazation" techniques for constraint satisfaction problems. Some have emphasized the suitability of these methods for parallel processing, while others have stressed the avoidance of thrashing. We feel that the chief advantage of these methods lies in their potential parallelism, especially since Gaschnig [13] has shown that more sophisticated backtracking methods can outperform sequential implementations of constraint filtering.

In order to perform constraint filtering, it is necessary that those nodes (regions) that constrain each other be connected in a network. It is at least theoretically possible for the labelling of a region to be influenced by any other region in the image, so ideally the constraint network should be a complete graph, connecting each region to every other region. But in practice this is not feasible, since the number of interconnections grows as the square of the number of regions -- far too fast. Having too many interconnections is undesirable for two reasons: First, it increases the cost of building a hardware network for constraint filtering (though some architectures, such as that of ZMOB [19], permit arbitrary interconnection at no extra hardware cost). Second, and more important, it increases processing time, since the amount of computation done for each region is roughly proportional to the number of regions it is connected to.

Therefore, it is desirable to limit the number of interconnections. This must be done carefully, since the correctness and effectiveness of the constraint filtering depend on the completeness of the interconnections, and the lack of a necessary connection might prevent or mislead the application of an important constraint. We would like to build a network of interconnections that is as sparse as possible, but still produces the same results as the complete graph. Obviously, this cannot be known ahead of time--the best we can do is to connect those regions that have a good chance of being relevant to each other. This is a matter that requires much further investigation, but for now we have implemented a simple notion of relevance: A region is connected to all regions that are very close to it (because these make up its immediate context), and to all very large regions in the image (because these give a good basis for judging it in its global context). Thus the number of interconnections is roughly constant for each node and overall is proportional to the number of regions in the image. This interconnection scheme is imperfect, but appears to work with few errors, at least for the domain of FLIR images used in this report.

Once the configuration of the network is complete, the constraint filtering proper can begin: We first of all attach to each region a list containing all the labellings that it might possibly bear. (Currently, these will be all labellings possible in the domain, although it should be possible to use context and the taxonomy of labels to reduce this initial list considerably.) Each label has associated with it a special "when-proposed" procedure, which is executed for each region whenever that label is first proposed for the region. This permits the calculation of certain parameters that make sense only if the region is interpreted as a particular sort of object. For example, if a region is hypothesized to correspond to an object of a certain intrinsic size, then it may be useful to use the region's apparent size, in conjunction with the camera geometry, to compute the object's range and location in space. Notice, however, that this computation makes sense only under this hypothesis.

Next, the label lists are filtered using what are called here "unary constraints". That is, knowledge about the intrinsic properties of objects is used to eliminate incorrect labellings from each region's label list. Regarded as propositions, the unary constraints have the form: "If a region is to bear this label, then the region must have these properties". The constraint is actually used in the contrapositive form: "If a region does not have these properties, then it cannot bear this label." These properties may be immediate properties of the region, or they may be those computed indirectly by the appropriate when-proposed procedure.

Clearly, these three steps (hypothesizing labels, computing parameters, and filtering labels) could be done at one swoop, with some improvement in computational efficiency. Here they are done separately, for clarity of presentation and ease of programming.

After the region labels have been filtered, we can attach to each interconnection (or arc) a list of label pairs that is the cross-product of the sets of labels on the two regions at either end of the arc. This list represents the joint labellings that are simultaneously possible for the two regions considered. Then all these label-pair lists can be filtered by binary constraints, that is, those joint labellings can be eliminated that violate a constraint on the labelling of pairs of regions. These constraints have the propositional form: "If two regions (say r_1 and r_2) are to simultaneously bear the labels ℓ_1 and ℓ_2 respectively, then r_1 and r_2 must stand in certain relations to each other." Again the constraint is used in its contrapositive form: If the two regions fail to stand in the required relations to each other, the appropriate pair of labels can be deleted from the arc joining them.

Following all this, three more filtering processes can be applied. One of them, filtering by existential constraints, enforces constraints of the following form: "If a region is to bear a certain label then there must exist other regions that have certain properties and stand in certain relationships with the given region." This is very much like a unary constraint, except that the properties of the other regions include the requirement that they bear certain labels, and that those labels are permitted simultaneously with the labelling to which the existential constraint is being applied. Thus existential constraints must examine the arc labellings. Unary constraints need be applied just once, but since the allowable labellings of arcs change during the constraint processing, an existential constraint that is satisfied early may later be violated because a labelling that it depended on has been rejected. Hence the filtering by existential constraints should be redone every time the arc labellings change.

The other two filtering processes, arc-upon-node interaction and node-upon-arc interaction, attempt to enforce consistency between the node labellings and arc labellings. The first process ensures that every node labelling has support from every arc that impinges on it. By "support" we mean that there exists on each arc at least one label pair that has the same label as the region for its first or second component, as appropriate, depending on which end of the arc the node lies at.

If a node labelling lacks support it is deleted. The second process ensures that every arc labelling has support from the nodes at either end of it. Here, by "support" we mean that every label pair on an arc should have as its first element a label that is represented in the labelling of the node at the appropriate end of the arc, and have as its second element a label that is represented in the labelling of the node at the other end of the arc. Any arc label that lacks such support is deleted.

These three interdependent filtering processes provide a simple but effective way of propagating inferences about the identification of regions through the constraint network. They provide the system with a rudimentary form of reasoning about scenes in the sense that its conclusions, if justified logically, would take several proof steps from the given axioms (these are the region properties and relations, and the domain constraints). Of course, all this reasoning is done by a mechanical process of propagating the effects of deleting node and arc labellings, but it can be regarded as a limited form of logical deduction.

Notice that all the filtering processes work strictly by refuting and eliminating labellings. This means that, after the initial labelling generation processing, all the filtering processes could be run independently and asynchronously on the regions in any order, without the fear of race conditions occurring. That is, the results of the constraint filtering will be the same, no matter in what order the individual filtering processes are applied to each node, provided all processes are applied until the network stabilizes--when no further deletion of labellings can be made. However, in the interests of efficiency and simplicity and in order to simulate an actual parallel implementation, we apply the various processes synchronously in parallel over the entire network. As described above, we first perform all the initial node labellings, next all the node filtering by unary constraints, then all the generation of joint labellings on arcs, followed by arc filtering by binary constraints. Now, the arc-upon-node interaction and the existential constraint filtering use the arc labellings to update the node labellings; and the node-upon-arc interaction updates the arc labellings using the node labellings. Therefore it is appropriate to apply these three propagation processes in a cycle of three (in the order given) repeatedly until the network stabilizes (when no further deletions of labellings can be made).

After the constraint filtering has stabilized and terminated, we can turn our attention to the interpretation of its results. Unfortunately, these results will not necessarily be correct in the sense of being a valid solution to the given constraint satisfaction problem. Ideally, we would like to see every region correctly and uniquely labelled with its identification as an object or object part. However, given the way we have decomposed the problem so as to make it amenable to parallel processing, such an outcome cannot be guaranteed. Before discussing these erroneous results in detail, we should stress that both in the example domain used here, and in other domains [17,20], we have not found these errors to be a serious problem in practice. Other authors [21], report similar findings.

One sort of error is that after the filtering a region may retain several labels, not just one. This situation can arise from two causes. First, it may well be that there is more than one valid solution to the constraint satisfaction problem, that is, the image admits of several distinct interpretations. So a given region may have more than one correct identification ascribed to it, either of itself, or in conjunction with multiple identifications of neighboring regions (neighboring in the sense of being directly connected in the network). From one point of view, this can hardly be considered an error: a region can have several different interpretations, and all of these are retained. But if a number of regions all have multiple labels, it may be of interest to discover which unique labellings of all of them are simultaneously possible, and this sort of unravelling cannot be done by mere constraint filtering. Related to this is the second cause of multiple labelling: There may exist in the network an ambiguity that can be resolved in principle, but cannot be resolved by pairwise constraint filtering--its resolution requires the simultaneous examination of the labellings of three or more nodes. Errors such as these are not serious, since they tend to occur infrequently--for most domains it seems that pairwise interaction is sufficient for essentially unambiguous interpretation. Even when they occur they can easily be resolved, by some sort of backtracking technique alone, or in combination with further constraint filtering, as used by Barrow and Tenenbaum in MSYS [21], and by Haralick and Shapiro [15,16]. In most cases, the bulk of the disambiguation will have been done by the constraint filtering, leaving very little work to be done by the final backtracking. However, we have not implemented such a post-processing phase for the current system because our main interest is in the filtering itself.

It is worth remarking that if only unary and binary constraint filtering are used, or existential constraints are used but the network is sufficiently complete, extra labelling (as discussed above) is the only sort of error that can occur. Under these circumstances constraint filtering will be safe in that it will never reject a correct labelling, even though it may retain some incorrect labellings. This means that if every region bears a single label, then we can be sure that all these labellings comprise the unique, correct solution to the constraint satisfaction problem. If any regions bear multiple labels, then we know that unique interpretations could be found, if necessary, by a later backtracking process. Unfortunately, if existential constraints are used, then the required completeness of the network cannot be guaranteed unless it is a complete graph, which is seldom feasible in practice.

The other sort of error that can occur is that a correct labelling is mistakenly rejected. As implied above, this can only happen when existential constraints are applied to a network that lacks some necessary interconnections. Since the completeness of the interconnections cannot always be determined ahead of time, it cannot be foreseen whether such errors will occur. If they do occur, they they are irreparable, since a label once lost cannot conveniently be reinstated. But once again, these errors, while theoretically possible, have not occurred in our examples because our simple configuration rule mentioned earlier ensures sufficient interconnection in the network--at least for the existential constraints used in the example domain.

Related to this is a problem that occurs if a region loses all of its labellings, that is, all possible identifications of it can be refuted. This means that the region is unrecognizable as anything from the assumed domain. But once any node in the network is unrecognizable, its effect will be propagated until all nodes lose all their labels. Strictly speaking, this is perfectly correct: If a scene contains objects that cannot be recognized, then the scene could not possibly be from our chosen domain, and therefore the whole scene is essentially unrecognizable. The problem is that the constraint filtering implicitly assumes that the set of labels and constraints correctly account for everything that might possibly appear. If this assumption is violated, then the entire image must be rejected, even though the image could be successfully interpreted if the alien object were not there. While theoretically justifiable, this behavior is undesirable in practice. If such a vision system were turned loose on the world, we would not want it to effectively go blind every time an unexpected object chanced into its field of view. One solution to this problem is to postulate a catch-all label for which there are no constraints whatsoever. Any region in the image can therefore bear this label, even those that are otherwise unrecognizable. Of course, all recognizable regions will also bear this label in addition, and there will be numerous additional label pairs attached to the arcs. This will cause no problem with the interpretation, but it does introduce a certain computational overhead which may not be negligible. Another solution, which does not suffer from these problems, is this: When a node loses all its labels, it should be marked as unrecognizable, and then removed from the network, with its connecting arcs as well, so that the undesirable effects cannot spread further. Both of these solutions have ramifications that we shall not go into here. Because of this, and because the problem only arises when the model embodied in the constraints is inadequate, we have not made any special provision for handling it in the prototype system. If any region is found to be unrecognizable, we go back and revise the model to account for the misrecognition.

This brings us to one final matter: How are the constraint models for a particular domain constructed in the first place? Winston [22] has proposed an automatic system for building scene models, that uses inductive inference over a set of training examples. Such an approach is certainly possible here, but the problem of separating relevant from irrelevant features can be expected to be very difficult for all but the simplest scenes. For now, we expect that models will be built by hand. A user of the system relies on his own introspection and knowledge of the domain to construct an initial model, applies it to some well-chosen examples, diagnoses any errors, and then corrects the model accordingly. For many applications, this is a quite acceptable way of building models.

In order to illustrate the matters presented above, we now give examples of the operation of our prototype system in a particular domain.

4. An example domain -- TANKSWORLD

In order to test out the ideas described in the previous sections, we have implemented a prototype system for scene analysis using constraint filtering, and applied it to a domain of forward-looking infra-red (FLIR) images of tanks and other military vehicles on fairly open ground. The image segmentation and region extraction programs were written in the programming language C. The constraint filtering system was written in LISP. This includes the constraint filtering procedures themselves, and also a number of auxiliary procedures, including those that provide the relational primitives out of which the constraints were constructed. The constraints were written as logical expressions in these primitives, using special conventions to mark the variables for the regions.

Example images from this chosen domain (dubbed "TANKSWORLD") can be seen in Figure 3. We admit five principal region labels in TANKSWORLD:

GROUND	(corresponding to the ground or any patch of ground)
SKY	(the sky or any patch of sky)
SMOKE	(a puff of smoke or similar bright compact object)
TANK	(a tank, or any vehicle)
TREE	(a tree or shrub)

Only TREE and TANK have any real size restriction, so only for these is oversegmentation a problem. Therefore we provide two additional labels, TANK-FRAGMENT and TREE-FRAGMENT, to cover pieces of these objects.

In general, the spatial position of an object cannot be determined from a single image. All that can be said is that the object must lie along a certain line of sight. However, we know that TANKS and TREES (we use the labels here informally to stand for the classes of objects they represent) must stand on the ground, and if we assume that the ground is an approximately level plane, we can use projective geometry and a simple camera model in order to fix their actual spatial locations, and from this determine their ranges, and their actual sizes from their apparent sizes in the image. The simple camera model used for these computations is shown in Figure 4. It assumes that the image is formed by a simple pin-hole camera, with known parameters. For objects in space we use Cartesian coordinates x,y,z, with the origin on the ground vertically below the camera's pinhole. The x axis runs along the ground to the right from this origin, the y axis directly forward, and the z axis vertically. In the image we use coordinates ξ (horizontal) and η (vertical) relative to an origin at the center of the field of view. These coordinates are related by

$$\frac{\xi}{f} = \frac{x}{y\cos\varphi - (z-h)\sin\varphi}$$

$$\frac{\eta}{f} = \frac{y\sin\varphi + (z-h)\cos\varphi}{y\cos\varphi - (z-h)\sin\varphi}$$

where h is the height of the pinhole above the ground, f is the distance from the pinhole to the film plane, and φ is the dip angle below the horizontal of the optical axis of the camera. These equations give an adequate approximation for any camera, provided the field of view is not too wide. In practice, these parameters should be known. For the images used here they were not known, but were estimated by taking measurements on the images of objects whose size was approximately known.

So for the labels TANK and TREE, we have a when-proposed function that computes spatial location on the ground and approximate vertical and horizontal extent. For the corresponding fragments the when-proposed function can compute only bounds on these values, but these bounds are nontheless useful.

There are 62 constraints used in the current model. The unary constraints are used to enforce the size restrictions on TANKS, TREES and their fragments, limits on the height to width ratio for TANKS and TREES, and restrictions on the position of SKY and GROUND relative to the horizon. The binary constraints are used in two ways: First, to express that the region for a compact object such as TANK, TREE, SMOKE cannot surround the region for any other sort of object (except that TANKS and TREES can surround their respective fragments). Second, to enforce restrictions on the relative brightness of objects, that SMOKE is brighter than anything else, TANK is not brighter than anything else, and that objects of the same class have roughly the same brightness, except for GROUND which has considerable variation. (Notice that the system is given no knowledge of the absolute brightnesses of objects—only relative brightness is used. This was done deliberately in order to demonstrate the ability of the constraint filtering.) Finally, the existential constraints capture the requirement that TREES and TANKS must rest upon a piece of GROUND, and that a fragment of an object must have next to it another fragment of the same sort such that the two taken together do not exceed the size restrictions for the corresponding whole object.

In Figure 5, we show some typical subimages from this domain. Figure 6 shows these images after segmentation with boxes drawn around the regions. Because of memory limitations of the present implementation, regions below a certain size were ignored, and interconnections were made only between regions whose boxes were immediately adjacent or overlapping, and to the one or two largest regions in each image. The constraint-filtering system was run on these examples, and the results are presented in Figures 7 and 8. (For each label, unambiguously labelled regions are shown in white; ambiguously labelled regions, which bear other labels as well, are shown in gray.) In all cases the constraint filtering stabilized after only a few iterations of the propagation processes.

As can be seen, the results are quite good, especially considering the noisiness of the original image, and the blind simplification done by the segmentation and the region extraction. A number of problems with the results are worth discussing, since they illustrate limitations of this approach.

Since there are so few constraints on GROUND, many other sorts of objects will retain this label. In a sense, this is perfectly unobjectionable. In these images there is no way of distinguishing a tank from a patch of ground with the same shape and coloration as a tank. In this domain the TANK interpretation is more likely, but the constraint filtering has no mechanism for expressing preference between two logically irrefutable labellings.

Another problem is that because there is often little contrast between sky and ground, quite a number of regions straddle the horizon, and thus admit both the labels SKY and GROUND. It is clear that the segmentation is wrong, but the current system can only accept uncritically the regions it receives from the segmentation. A more sophisticated system could attempt to modify the segmentation when such a contradiction was detected. In a few cases, an object clear to the eye is merged with another object because of a short segment of low contrast boundary between them and is thus lost altogether. Detecting and repairing such a mistake in the segmentation is really quite difficult.

The limited context provided by the limited interconnection of regions causes some difficulties. There are a few regions that retain the label SMOKE, not because they are the brightest regions in the image, but merely because they are brighter than anything they are connected to. In some other images it happens that a cluster of TREE-FRAGMENTS support each other, even though altogether they are too large or too small to comprise an entire TREE. The system takes into account only the pairwise interactions of the fragments, without trying to organize them into a coherent whole.

There are some other misidentifications that can be blamed on the simplified shape description used here. A number of odd-shaped regions are labelled as TREES just because the regions happen to fix boxes of about the right size and shape, even though it is apparent that they look nothing like TREES in their actual shape.

Despite these problems, it is clear that the constraint filtering can accomplish almost all the task of analyzing these scenes. In the next section, we will discuss some of the issues raised by these problems, and consider ways of extending the constraint filtering process to overcome them.

5. Discussion

We have seen in the previous section that constraint filtering is a feasible technique for scene analysis, even if implemented in a very simple way. We will discuss here some extensions to this technique that would overcome most of the shortcomings of the current approach, and lead to a more powerful and flexible scene analysis system.

One straightforward improvement would be to provide a more accurate shape description for regions, which would permit more realistic computation of region properties and relations. The representation of regions by boxes is convenient, but hardly satisfactory. In a few cases, a spurious adjacent or surround relation will hold between the boxes of two regions, when in fact it is not true of the regions themselves. This can lead to errors in interpretation.

It would be desirable to provide some facility for indicating preference between several labels for a region, all logically equally possible, but one far more likely. The unavoidable labelling of TANKS also as GROUND, mentioned in the previous section, illustrates this problem. More generally, as suggested by numerous authors [14,21,23], it would be useful to attach probabilities or confidence measures to all the hypotheses, properties and relations in the system, and provide a calculus for combining these confidence measures. As an example, the relation same-brightness just checks that the difference in brightness between two regions is below some given threshold. In most domains this is unsatisfactory. There is no sharp cut-off between "same" and "not the same". All we can say is that the greater the difference in brightnesses between two regions, the less reasonable it is to regard them as having the same brightness.

Related to this is the need for a more subtle combination of evidence. A certain label may have a number of constraints applicable to it. If a certain region passes all but one of these constraints it would lose that label. But in some circumstances it may be more reasonable to suspect that the labelling is correct but that some error has been made in the evaluation of the failed constraint. Perhaps an important piece of evidence was obliterated by noise, occlusion, or poor segmentation. Ideally a scene analysis system should be able to tolerate such lost evidence, and even attempt to recover it by a closer re-examination of the original image.

This brings us to the matter of the interaction between the scene analysis system and the image data. In the current system there is a strictly one-way flow: segmentation, then analysis of the segmentation. It would be preferable to have a mechanism whereby the higher-level analysis could, under certain conditions, call for a re-examination of parts of the original image in order to search for features that may have been lost in the initial processing. The scene analysis system should also be provided with an arsenal of assorted image processing routines, in addition to segmentation, in order to capture lines, spots, and other features that are likely to be lost during segmentation.

The current scheme for interconnecting regions into a network, while effective, is fairly ad hoc. It should be possible, by an analysis of the constraints, to make a more rational decision about the connection of regions. A region may need only to be connected to regions that stand in certain relations to it and that retain certain labels after the unary constraint filtering, for these are the only regions that could possibly falsify the applicable constraints. More generally, it may be advantageous to permit the reconfiguration of the network during processing, although this would have to be done with great care in order to retain the desirable properties of constraint filtering.

One deficiency of the current system is its clumsy notation for expressing constraints that apply to a number of labels. As mentioned earlier, to express the notion that SMOKE is the brightest object in the domain we must provide a separate brighter-than constraint for every other label in the domain. This could be overcome, at some cost in efficiency, by permitting some sort of quantification over labels, allowing constraints like "For all labels ℓ, not equal to SMOKE, SMOKE is brighter than ℓ." But this is only a cosmetic change, which does not address the underlying problem. The current system regards all the labels as being quite independent of each other. This becomes quite a serious computational inefficiency as the number of labels becomes large (as it will for any realistic domain), especially as the amount of calculation on each arc is roughly proportional to the square of the number of labels in the domain. But in reality, the labels in a particular domain will usually show certain similarities among themselves and share many constraints. It is wasteful to independently re-evaluate for each label these shared constraints. This inefficiency can be naturally and effectively overcome by organizing the labels of a domain into a taxonomy based on similarity and shared constraints. The system could initially propose generic labels, which stand for whole classes of objects, and test these labels by applying only those constraints common to whole classes of objects. Later, when no further progress could be made by such general reasoning, the generic labels could be replaced by more specialized labels, and more specialized constraints could be applied. For example in TANKS-WORLD, we could group all objects that must lie below the horizon into a single class, and eliminate this class label from all regions that lie above the horizon. Once this had been done we could specialize objects below the horizon into classes of compact and extended objects. Later, the compact objects could be subdivided into TANKS and TREES. If necessary, TANKS and TREES could be further classified into their different models and varieties. While this scheme is intuitively clear, some work still remains to be done in order to properly formalize it, especially in regard to the interactions between nodes at different levels of specialization.

The current system also suffers from a one-level treatment of network nodes. It is possible for a cluster of regions to retain the label TREE-FRAGMENT, even though the cluster, considered as a unit, looks nothing like a TREE. What is needed is a mechanism for creating new nodes having existing nodes as parts. This becomes more acutely necessary in more complex domains whose objects may be built up from distinct parts. Even in TANKSWORLD, at slightly better resolution, a TREE would be seen to consist of a trunk, branches and foliage; and a TANK would show wheels, turret, gun-barrel and other details. Such techniques for hierarchical constraint filtering have been studied in simpler domains [24,25], but require further development for more complex domains, especially if they are to be applied in an efficient manner.

Recently, Davis [26] has shown that constraint filtering, expressed formally in logic, can be regarded as a limited form of inferencing. This raises the possibility that more powerful forms of constraint filtering could be devised. Currently, these techniques work by falsifying simple hypotheses about the individual and joint identifications of nodes in network. More powerful techniques could conceivably reason about other properties and relations between nodes, for example, occlusion relations between objects. Formal logic and theorem-proving would also provide a convenient means of treating some of the other extensions of constraint filtering described above.

In conclusion, we have shown that constraint filtering is an effective means of scene analysis in a domain more complex than has previously been used with such techniques. The deficiencies of the approach, as revealed by the results we have obtained, have suggested a number of improvements and extensions to constraint filtering. The development of these extensions is the object of current research.

REFERENCES

1. H. G. Barrow and J. M. Tenenbaum, "Recovering intrinsic scene characteristics from images," pp. 3-36 in Computer Vision Systems, A. R. Hanson and E. M. Riseman, eds., Academic Press, New York, 1978.

2. H. G. Barrow and J. M. Tenenbaum, "Representation and use of knowledge in vision," SIGART, no. 52, June 1975; also SRI AI Center Technical Note 108, July 1975.

3. D. Marr, "Early processing of visual information," Phil. Trans. Roy. Soc., vol. B275, 1976, pp. 483-524; also MIT AI Lab Memo 340, Dec. 1975.

4. J. M. Tenenbaum and H. G. Barrow, "Experiments in interpretation-guided segmentation," Artificial Intelligence, vol. 8, 1977, pp. 241-274.

5. J. A. Feldman and Y. Yakimovsky, "Decision theory and artificial intelligence: A semantics-based region analyzer," Artificial Intelligence, vol. 5, 1974, pp. 349-371.

6. M. Nagao and T. Matsuyama, A Structural Analysis of Complex Aerial Photographs, Plenum Press, New York, 1980.

7. K. A. Narayanan and A. Rosenfeld, "Image smoothing by local use of global information," IEEE Trans. Syst. Man, Cybern., vol. SMC-11, 1981, pp. 826-831.

8. H. G. Barrow and R. J. Popplestone, "Relational descriptions in picture processing," pp. 377-396 in Machine Intelligence 6, B. Meltzer and D. Michie, eds., University of Edinburgh Press, 1971.

9. H. G. Barrow, A. P. Ambler and A. M. Burstall, "Some techniques for recognizing structures in pictures," pp. 1-29 in Frontiers of Pattern Recognition, S. Watanabe, ed., Academic Press, New York, 1972.

10. D. Waltz, "Understanding line drawings of scenes with shadows," pp. 19-92 in The Psychology of Computer Vision, P. H. Winston, ed., McGraw-Hill, New York, 1975.

11. A. K. Mackworth, "Consistency in networks of relations," Artificial Intelligence, vol. 8, 1977, pp. 99-118.

12. J. Gaschnig, "A general backtrack algorithm that eliminates most redundant tests," 5th IJCAI, Cambridge, MA, Aug. 1977, p. 457.

13. J. Gaschnig, "Experimental case studies of backtrack vs. Waltz-type vs. new algorithms for satisficing(sic) assignment problems", Proc. Canadian Soc. for Comp. Studies of Intelligence, Toronto, July 1978, pp. 268-277.

14. A. Rosenfeld, R. Hummel and S. Zucker,"Scene labeling by relaxation operations," IEEE Trans. Syst. Man Cybern., vol. SMC-6, 1976, pp. 420-433.

15. R. M. Haralick and L. G. Shapiro, "The consistent labeling problem: Part I," IEEE Trans. Patt. Anal. Mach. Intel., vol. PAMI-1, 1979, pp. 173-184.

16. R. M. Haralick and L. G. Shapiro, "The consistent labeling problem: Part II," IEEE Trans. Patt. Anal. Mach. Intel., vol. PAMI-1, 1979, pp. 193-203.

17. L. Kitchen and A. Rosenfeld, "Discrete relaxation for matching relational structures," IEEE Trans. Syst. Man Cybern. vol. SMC-9, 1979, pp. 869-874.

18. J. J. McGregor, "Relational consistency algorithms and their application in finding subgraph and graph isomorphisms," Information Sciences, vol. 19, 1979, pp. 229-250.

19. C. Rieger, "ZMOB: Doing it in parallel," Proc. IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Hot Springs, VA, November 1981, pp. 119-124.

20. L. Kitchen and E. V. Krishnamurthy, "Fast, parallel, relaxation screening for chemical patent data-base search," to appear in Journal of Chemical Information and Computer Sciences.

21. H. G. Barrow and J. M. Tenenbaum, "MSYS: A system for reasoning about scenes," SRI AI Center Tech. Note 121, April 1976.

22. P. H. Winston, "Learning structural descriptions from examples," pp. 157-210 in The Psychology of Computer Vision, P. H. Winston, ed., McGraw-Hill, New York, 1975.

23. L. Kitchen, "Relaxation applied to matching quantitative relational structures," IEEE Trans. Syst. Man Cybern., vol. SMC-10, 1980, pp. 96-101.

24. L. S. Davis and A. Rosenfeld, "Hierarchical relaxation for waveform parsing," pp. 101-109 in Computer Vision Systems, A. R. Hanson and E. M. Riseman, eds., Academic Press, New York, 1978.

25. L. S. Davis and T. C. Henderson, "Hierarchical constraint processes for shape analysis," IEEE Trans. Patt. Anal. Mach. Intel., vol. PAMI-3, 1981, pp. 265-277.

26. L. S. Davis, "A logic model for constraint propagation," Tech. Report 137, Computer Sciences Dept., University of Texas, Austin, TX, Feb. 1980.

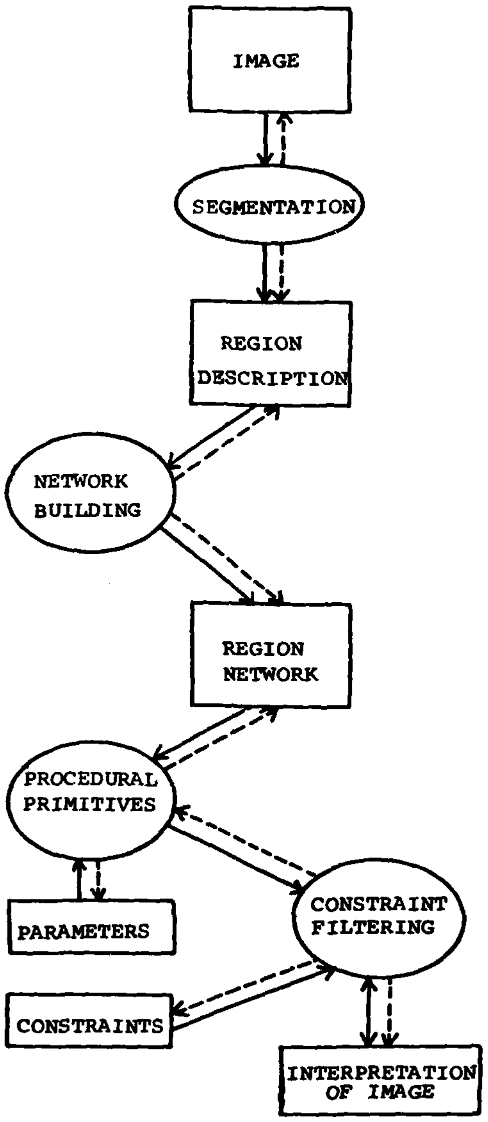
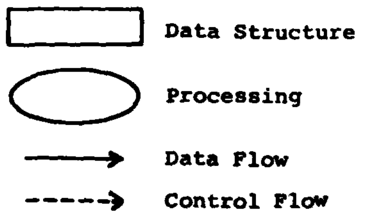


Figure 1. Informal overview of system



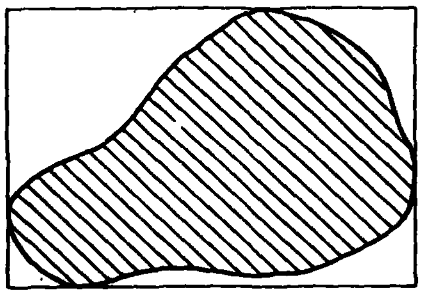


Figure 2. Circumscribing upright rectangle ("box") used to describe a region

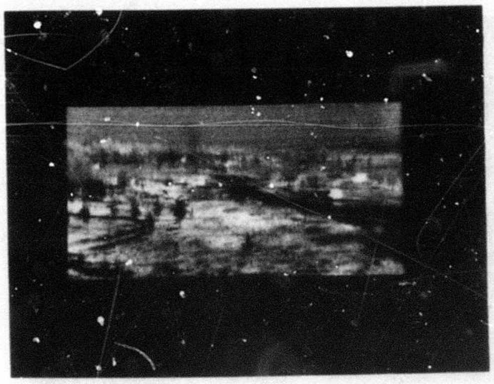
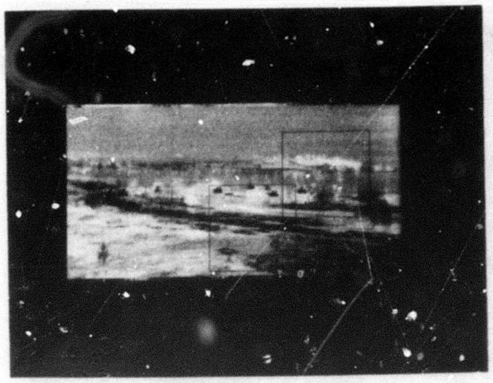



Figure 3. Three typical FLIR images.

a c

b

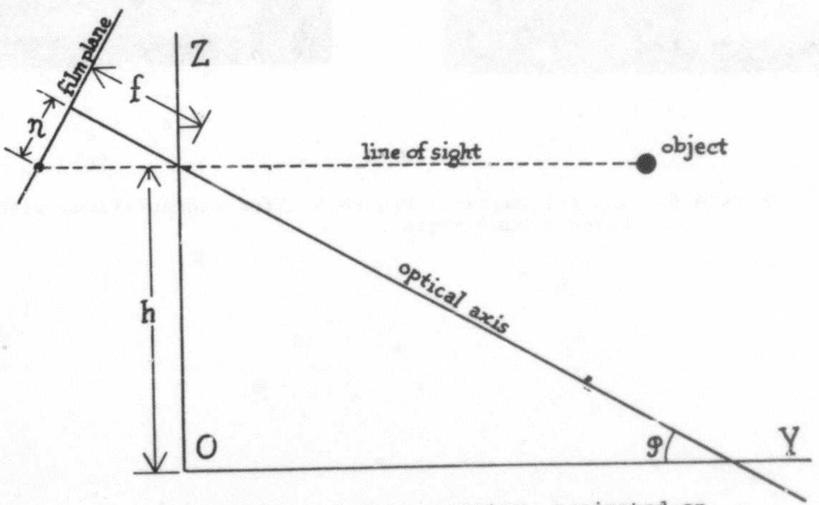
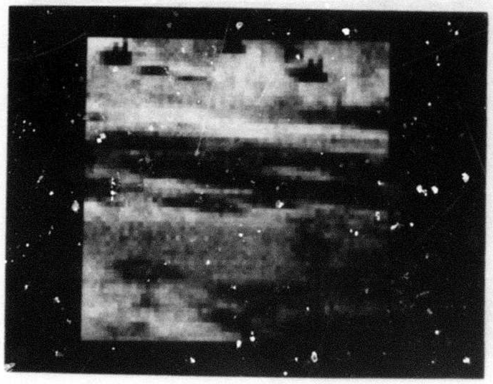
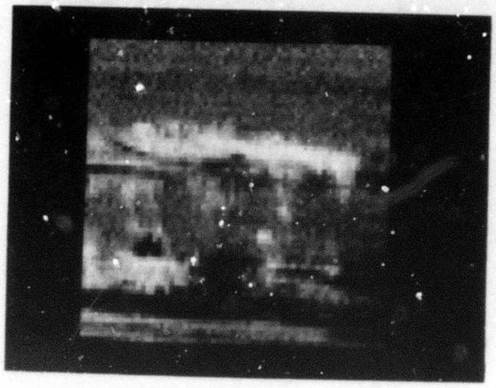


Figure 4. Camera geometry, projected on plane YOZ. x and ξ axes not shown

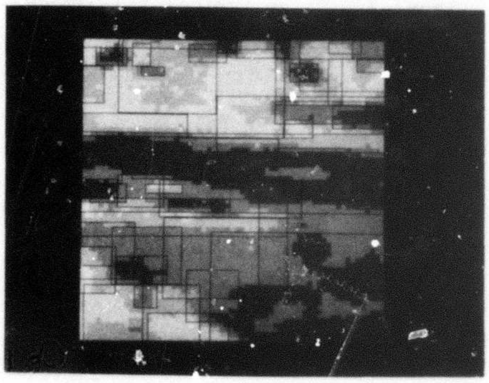


(a)

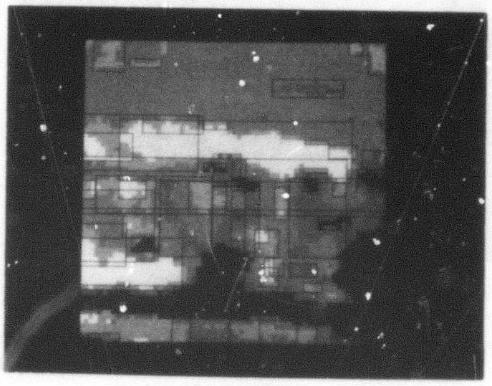


(b)

Figure 5. Two FLIR subimages (boxed in Fig. 3c)



(a)



(b)

Figure 6. The subimages in Figure 5 after segmentation, with boxes around regions.

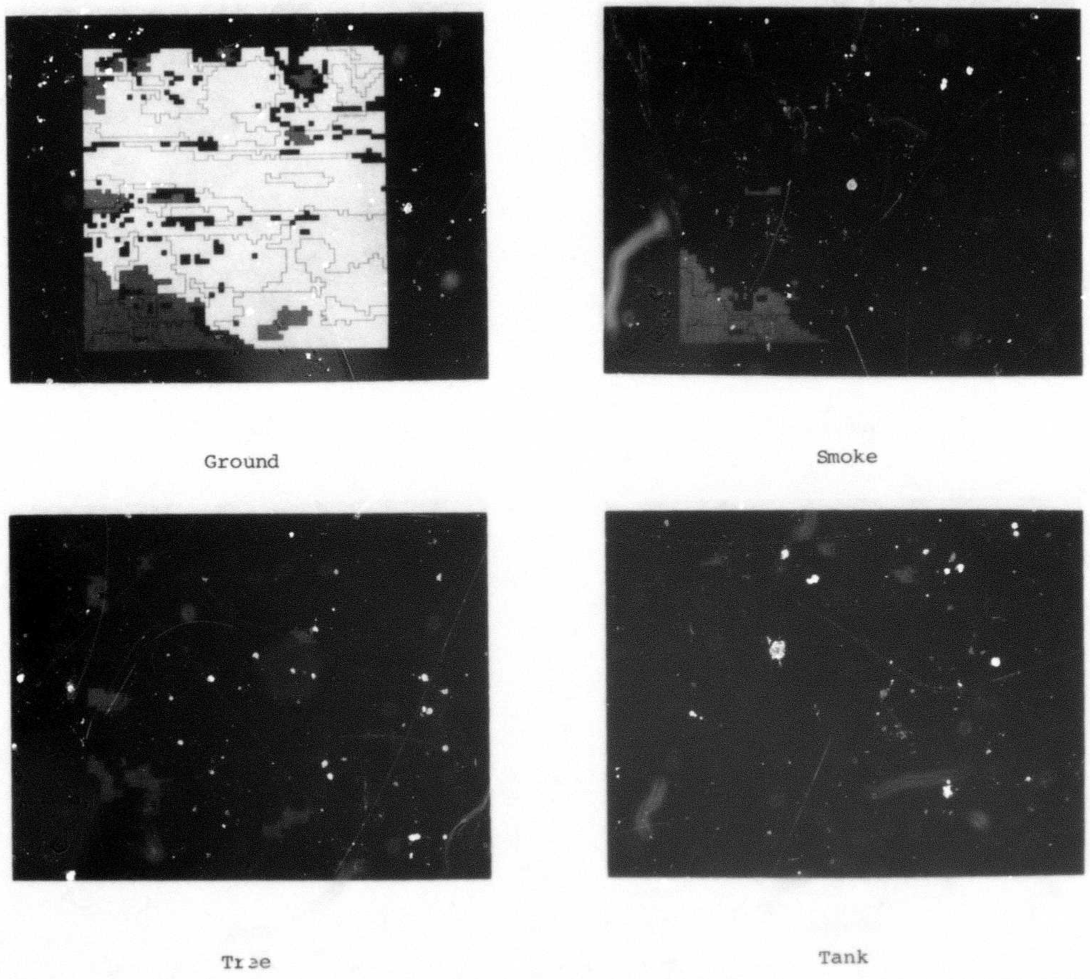


Figure 7. Results of constraint filtering for Fig. 5a.

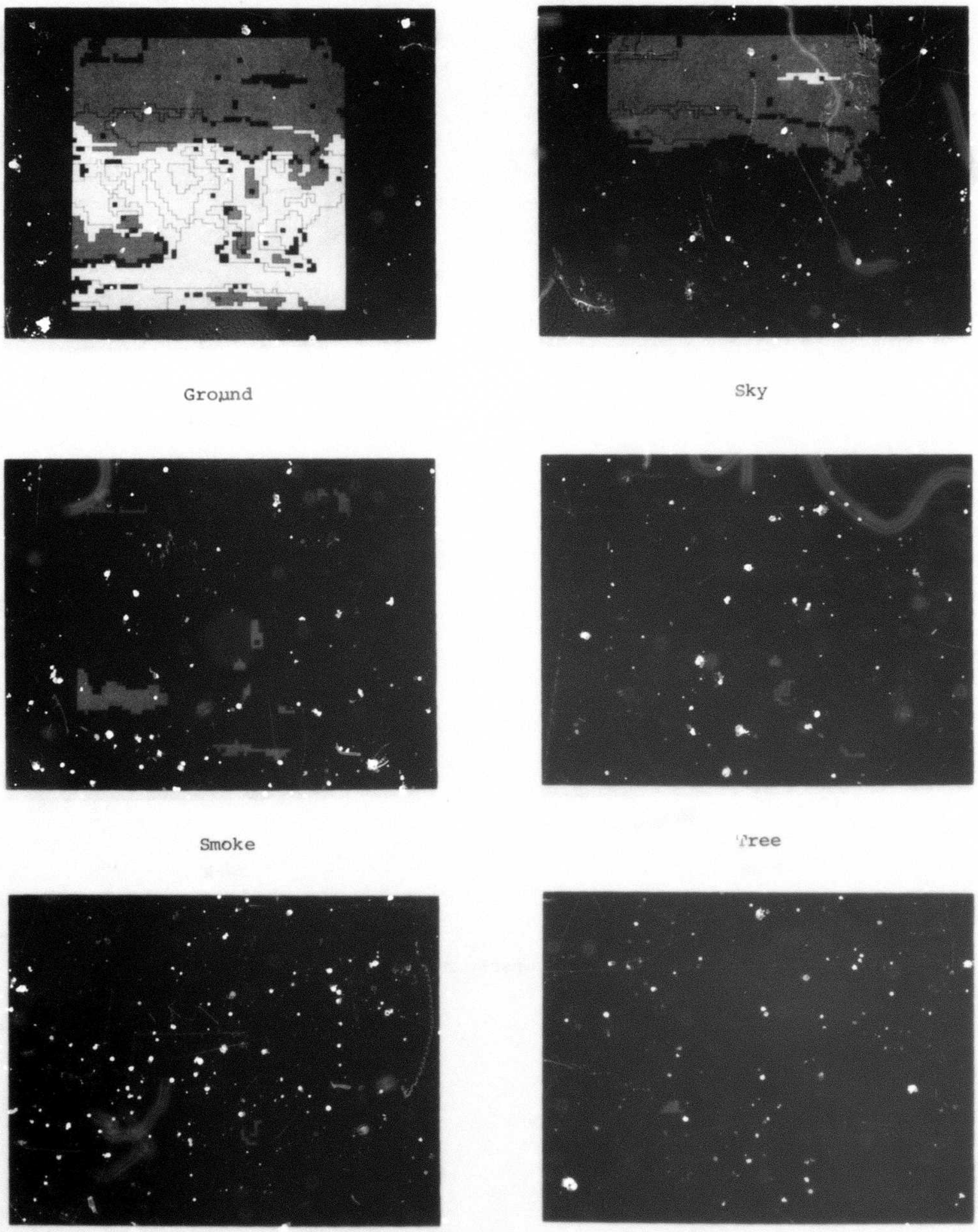


Figure 8. Results of constraint filtering for Fig. 5b.

AD-P000130

STRUCTURAL TEXTURE ANALYSIS APPLICATIONS

F. Vilnrotter* and R. Nevatia
Departments of Computer Science
and Electrical Engineering
University of Southern California
Los Angeles, Ca. 90089-0272

1. INTRODUCTION

This paper presents a discussion of two applications of the results of a structural texture analysis system [8]. The details of the basic analysis procedure will not be presented in this report. The procedures to generate basic descriptions of natural textures are described in [1-2]. The extraction and description of texture elements is described in [3]. The final determination of relations (placement rules) and the reconstruction of texture patterns is presented in [4-5]. Short descriptions of these programs can also be found in [6-7].

The goal of the symbolic analysis program was to produce a description of a texture pattern which is similar to the type of descriptions produced by human observers. The description is based on the appearance and placement of elementary texture primitives. These descriptions can be used for reconstruction of the pattern, for recognition of the input pattern, and for analysis of texture gradients for determination of surface orientations. The last two topics are described in this report.

2. TEXTURE RECOGNITION

A texture recognition algorithm that uses the descriptions generated by our texture analysis programs is discussed below. The descriptions consist of the periodicity of the texture and the size and shape of the texture elements. Details of descriptions are given in part in [1-3,6] and all details may be found in [8]. Hopefully, the discussion of the recognition algorithm below is self-explanatory in terms of the descriptions used. The recognition scheme is basically a decision tree. Eleven types of textures were used in our experiments.

Texture Recognition Algorithm

The structure of the algorithm used for texture classification is shown in Fig. 1. The texture classified are: floor grating (dark dot pattern), brick wall, aerial view of city, raffia (woven palm), herringbone material, wood grain, aerial view of water, straw, grass, sand, and wool.

Most of the textures used are from the Brodatz album [12]. The exceptions are the floor grating, brick wall, and aerial city patterns. Aerial city pictures taken at different orientations and different scales were used. Only 11 samples are found. All other texture groups consist of 16 samples each. Pictures of both shifted and unshifted brick patterns are used.

The decision tree form was chosen for this texture classification scheme. The structure of the tree allows the classification information to be weighted by means of relative ordering. For example, the test for periodicity is encountered at an earlier stage than any of the aspect ratio tests. Clearly texture period information is being given more significance than aspect ratio information according to this classification scheme.

The first texture characteristic considered is periodicity. If a texture is non-periodic it will be classified by the subtree to the right of the root node. However, if the texture exhibits signs of periodicity but fails the tests for each of the 5 regular texture patterns it is sent back to the root node, re-labeled as a non-periodic texture. Due to this loop in the structure, the classification algorithm does not have the exact form of a binary decision tree. However, for convenience tree terminology will be used in the discussion. The loop is introduced to accommodate textures which are basically non-periodic, but may show evidence of periodicity some of the time. The wood grain, water and straw textures exhibit this characteristic. The opposite is not as likely to occur, i.e., the periodic textures are not mistakenly sent down the non-periodic branch.

No absolute texture element dimensions or intensity values are used; and all directional information is relative as well. In this way, the analysis should be insensitive to scaling, rotation and degree of contrast within the image. Measures which are used are primitive eccentricity, dimension to period ratio, the number and significance of texture element types, and relative intensities and orientations of texture primitive types. The details of each decision box for the two main decision branches is given below:

Periodic Branch

1) Dark Round Primitives - More than four dark primitives are merged into one primitive type.

*F. Vilnrotter is currently with Hughes Research Laboratory, Malibu, Ca.

2) Multiple Sized Primitives - Multiple sized primitives are found in two perpendicular directions. The texture period is equal to the sum of the element sizes. The sum of the (size/period) ratios for the four most significant primitives is less than .2.

3) Most Significant Primitives are Periodic in the Same Direction - The two most significant primitives exhibit an element spacing value for the same scan direction.

4) Large (Dark Element Size/Light Element Size) Ratio - The dark primitive is at least twice as wide as the light primitive in the most significant scan direction.

5) Light Elongated Primitives - The ratio of the dimension in the most significant scan direction to the dimension in the direction perpendicular to the direction of scan is less than .4.

Non-Periodic Branch

1) Uni-Directional Texture - The two most significant primitives are found in the same scan direction. The significance numbers associated with these two primitive types are greater than all other significance numbers by at least .66 (out of a possible 1.0).

2) Less Elongated Primitives - The sum of the aspect ratios for the two most significant primitives is at least as great as .175.

3) Most Significant Primitive is Relatively Dark.

4) Elongated Texture Primitives - The minimum aspect ratio of the four most significant texture primitives is no larger than .18.

5) Two Primitives Types - Primitives are found for the two most significant (intensity,direction) pairs.

6) Gradual Loss of Significance - There is no abrupt loss of significance after the fourth (intensity,direction) pair. The difference is smaller than .3.

7) Primitives Found in Two Perpendicular Directions.

8) Low Size/Spacing Ratio - There is a low (element size/element spacing) ratio for relatively dark primitives in two perpendicular directions. The sum of both ratios is less than .53.

9) Less Gradual Loss of Significance - The loss of significance after the fourth (intensity,direction) pair is at least .3.

No attempt has been made to optimize this decision scheme. The classification results are discussed in the following section.

Classification Results

Classification results are given in the confusion matrix shown in Table 1. The types of samples to be classified are listed to the left of the matrix. Each row shows how a specific set of samples was classified. For example, 15 aerial water texture samples were correctly classified, while one sample was incorrectly classified as wood grain. One hundred seventy-one samples were classified in all. In most cases the samples came from 512 x 512 pixel texture images. These were divided into sixteen 128 x 128 pixel non-overlapping texture subwindows. However, in the case of the aerial city samples only 11 samples were available. These were cropped from two different satellite images of the San Francisco area. The 16 brick wall samples were taken from three separate brick wall images. One hundred fifty-six samples were correctly classified to give an overall success rate of 91.23%. It should be noted that additional contextual information, e.g., color, scale, and type of scene would probably have improved the results obtained. However, no information of this type was used in the classification scheme in order that the strength of the texture descriptions used could be tested in isolation.

There are no mismatches for the highly structured, regular texture group. However, there are a number of non-periodic texture samples which are classified incorrectly. One source of confusion is between the water and wood grain textures. Both are one-dimensional textures made up of elongated texture primitives. The wood grain primitives tend to be more elongated than the water wave primitives. There is little else which is noticeably different from a structural point of view. Hence, confusion of these two texture types is predictable. Additional contextual information, e.g., the scale information for both textures, would improve the classification results.

Another area for confusion is the set of textures consisting of (grass, sand, and wool) exhibit the least amount of structure of the entire set. The edge images, ERAs, ERA descriptions, composite texture primitive masks, and texture primitive descriptions are very similar for all three types. This is because the program is not designed to measure the types of features which most readily differentiate these textures. In light of these description similarities, confusion among members of this group is to be expected. It should be noted that none of these texture samples were confused with textures from outside of the group and none of the other texture samples were mistakenly classified as one of these. The same can be said of the subgroup formed by the water and wood grain textures.

In summary 156 samples out of 171 were correctly classified to give an overall success rate of 91.23% for this classification scheme. These results are extremely encouraging. It would seem that the information extracted by the algorithms presented earlier in this thesis describe meaningful texture characteristics.

3. SURFACE ORIENTATION ANALYSIS

In Figs. 2 and 3 two texture gradient images are shown. Figure 2 is an image of a brick wall and Fig. 3 is an image of a redwood shake roof. In each case the textured surface is at a non - zero angle with respect to the image plane. The brick wall recedes to the left of the image, while the shake roof slants away from the viewer toward the top of the image.

One cue which we use to infer information about the orientation of textured surfaces is the texture gradient, i.e., the relative change in size or period of the elements making up the textured region within the image. Assuming that the textured surface being viewed is homogeneous, the element sizes of like texture primitives should decrease with increased distance from the viewer. The direction of maximum rate of change of depth with respect of the observer should be discernible as well as the degree of surface slant in this direction. A convenient way to represent these two surface characteristics is via the gradient space.

Let $z = f(x,y)$ be a function defining a planar surface in 3-Space, where the image plane is parallel to the x-y plane, (see Fig. 4). The surface normal, N, can be defined as follows: $N = (f_x, f_y, -1)$. Letting $p = f_x$, and $q = f_y$, we have the gradient vector, $G = (p,q)$. The direction of G is $\mathrm{TAN}^{-1}(q/p)$, while the magnitude of G is $\mathrm{SQRT}(p^2 + q^2)$. The direction of G denotes the direction of the greatest rate of change within the image, while the magnitude of G determines its quantity. Also, the tangent of the angle that the surface makes with the x-y (or image) plane is equal to the magnitude of G. Therefore, the orientation of a surface in 3-Space can be represented as a point in the gradient space.

In [10] Stevens suggests an alternate set of coordinates to represent surface orientation. He suggests the pair (σ,τ), where

$$\sigma = \tan^{-1}((p^2 + q^2)^{1/2}),$$

and

$$\tau = \tan^{-1}(q/p).$$

Stevens' slant and tilt angle terminology will be used. The tilt angle, τ, is an angle made with the horizontal axis of the image plane. It is the projection of the surface normal onto the image plane. Stevens has shown that this direction coincides with the direction which exhibits the greatest rate of change of distance from the observer to the surface. This is precisely the direction of the texture gradient. Therefore, by calculating the texture gradient one can determine the tilt angle. The slant angle, σ, is the angle which defines how much the image plane orientation differs from the surface plane orientation. For a discussion of various forms of surface orientation representation see the works of Stevens [10] and Kender [13].

Calculating the tilt angle is fairly straightforward. It entails determining the direction of the gradient of any texture measure which is scaled (due to distance), foreshortened (due to surface orientation), or both scaled and foreshortened.

Determining the slant angle, σ, is more involved. One method suggested is to determine which texture measure corresponds to the characteristic dimension. That is, which texture measure is scaled but not foreshortened. The normalized gradient of this dimension, taken in the direction of the texture gradient, is equal to the tangent of the slant angle.

$$\frac{\nabla d}{d} = \tan \sigma, \qquad (1)$$

where d is the characteristic dimension. Characteristic dimensions are parallel to the image plane and are perpendicular to the local surface tilt. Therefore, after the surface tilt has been determined the orientation of the characteristic dimension is known. This scheme for calculating cannot be used if the image is an orthographic projection, or if the elements exhibit successive occlusion. Alternative schemes are explored in [10] for handling these problems. Here it will be assumed that neither problem exists.

In [9] Bajcsy presents a method for calculating the angle formed by the image and surface planes. However, the dimension used in this case is the dimension oriented in the gradient direction. Therefore, it is both foreshortened and scaled.

Using the principles of projective geometry, the trigonometric rules pertaining to similar triangles and some small angle approximations, Bajcsy derives an expression for , the angle formed by the surface and image planes.

$$\frac{-\tan \alpha}{\text{Focal Distance}} = \frac{\text{Fractional Change in Element Size}}{\text{Baseline Image}} \qquad (2)$$

where the fractional change in element size and the baseline within the image are both measured in the direction of the texture gradient. Details of the derivation can be found in [8].

Both methods assume proximity to the line of sight defining the local image plane. All of the examples used in the following section have maximum off center angle less than 10 degrees. Further work is necessary to define the transformation needed to correct for large off center angular separations. This transformation should take into account the image plane and lens system characteristics as well as the geometry needed for coordinate transformations.

A General Orientation Analysis Technique

Application of our structural texture analysis techniques to surface orientation analysis problem is in a preliminary stage. The process has not been fully defined or automated. An outline of a proposed algorithm is discussed below, and some preliminary results are presented in the following section. Although only part of the program is operational, see 2 below, most of the remaining program sections are defined and seem feasible from a programming point of view.

One effect which must be anticipated by this technique is the possible changing orientation of the texture primitives. If the texture gradient was strong enough the height of the brick primitives in Fig. 2 would be found in the 120 degree scan direction in the lower left hand part of the image and in the vertical scan direction in the right and central areas. One possible solution is discussed in (3) below.

A possible scenario for detecting surface orientation is as follows:

1. Divide the textured region into locally uniform subwindows, i.e., subwindows which exhibit very little, if any, element size variation. This algorithm is not yet defined. However, as a first approximation the image can be divided into subwindows which accommodate the largest texture elements. Then element sizes can be averaged over each texture subwindow to produce an element size for that location in the image. This was done manually for the two examples discussed below. (In order to automatically determine the largest element size, one might calculate modified ERAs (see 2 below) for a large range of distances, say up to one third of the largest image dimension, over the entire texture image. Then an ERA interpretation routine, similar to the one presented in [2], can be used to determine the largest texture element dimensions.)

2. Calculate modified ERAs for each subwindow within the region. Only the first match encountered will be recorded for a particular directional scan. Hence, no element size or spacing repetitions should be counted. It is hoped that this will prevent repetitions from being interpreted as element size variations within a given subwindow. The modified ERA calculation scheme is operational and has been used to produce the results shown in Fig. 5.

3. Look for a dimension exhibiting strong results for each subwindow. A dimension, d, which is locally uniform but which reflects the gradient of the texture is sought. It should be verified that all of the ERA results chosen refer to the same texture primitive type. One way to achieve this end is to extract texture primitives for overlapping subwindows and verify that enough of the primitives extracted belong to both subwindow neighbors. In this way textural elements which shift orientation due to the effect of angular perspective can be found in each subwindow.

4. Create a matrix, M, of the centroid values of the element size peaks for d. Use a gradient operator on M to calculate the value of the tilt, or gradient, angle of the surface within the image.

5. Knowing the tilt angle means that the orientation of the characteristic dimension is also known. It can then be ascertained if either the characteristic dimension or the gradient dimension is already available as part of the set of ERA results. If this is the case then the rest of this step can be skipped. If this is not true then the characteristic or gradient dimension of some non-background texture primitive type must be measured from texture primitive masks. (These measurements should proceed outward from the elemental centers of mass.) Either of these two dimensions can be used for the slant angle calculation. Histograms of these dimensions should be kept so that the final calculations can be made using the highest amplitude/lowest standard deviation results. When measuring the characteristic dimension the original set of subwindows can be used. However, when calculating the gradient dimension the subwindows might have to be recropped to provide the correct center to center angle.

6. At this point either Eq. 1 or 2 can be used to calculate the surface slant angle to complete the procedure. If the characteristic dimension is known then Eq. 1 would be used. If the dimension oriented in the texture gradient direction is known then Eq. 2 would be used. Both slant angle calculation methods are discussed above.

Texture Gradient Examples

In this section two texture gradient examples are presented and discussed. They make use of the general method outlined in above.

Example 1

Consider the brick wall image in Fig. 2. This image is 512 x 512 pixels. It was divided into 16 128 x 128 pixel subimages, and ERAs were calculated for each of these. The vertical element size ERAs for each subwindow are shown in Fig. 5. As expected the element sizes decrease toward the left side of the image. Figure 6(a) shows the brick vertical element size dimension matrix. In Fig. 6(b), the results of the gradient calculation are shown; and Fig. 6(c) shows the results of the tilt and slant angle calculations. In this case the vertical brick dimension is the characteristic dimension, hence, the method developed by Stevens will be used to calculate the surface slant. The slant and tilt calculations are carried out for the 4 interior subwindows of the image (Fig. 2). The tilt angle measured from the image is approximately 3°. The tilt angle results range from $.967^{\circ}$ to 4.52°. Unfortunately, the actual slant angle is not known for this image. However, the results found for this angle seem to be reasonable. The angle made by the image and surface planes, or similarly, the angle between the principle ray and the surface normal seem to be in the neighborhood of 45°. In the next example the approximate angle formed by the surface and the image plane is known.

Example 2

Consider the shake roof image in Fig. 3. This image is 512 x 512 pixels. It was divided into 9 170 x 170 pixel subimages. (The last two rows and columns were not used.) ERAs were calculated for each of the nine subimages. The vertical element size ERAs for each subwindow are shown in Fig. 7. As expected the element sizes decrease toward the top of the image. Figure 8(a) is the matrix of the centers of mass for the vertical element size ERAs of Fig. 7. In Fig. 8(b) the results of the gradient and tilt angle calculations are shown, and Fig. 8(c) shows the results of the slant angle calculation for three image locations. The gradient direction dimension, i.e., the vertical dimension, of the wood shake elements was already known via ERA calculation. Therefore, the the surface slant angle was calculated using the method developed by Bajcsy. (It would be very difficult to use the characteristic dimension in this example since the widths of the wood shake are variable.) The slant angle computations are carried out for 3 sets of data. The tilt angle is calculated once since there are only enough windows for one gradient computation. The tilt angle for the image (Fig. 3) is approximately 90°. The tilt angle was computed to be 88.49°. The slant angle was calculated to be approximately 71.97°. (See Fig. 9.) The slant angle calculation results range from 72.405° to 74.167°.

Determination of window size is a problem which must be addressed. One possible solution is to calculate a set of ERAs for the entire image, initially, for a wide range of distances. The maximum element sizes found would then dictate the appropriate window size.

4. SUMMARY AND CONCLUSIONS

Structural texture analysis techniques previously developed were applied to 2 texture analysis problems, namely, texture recognition and surface orientation determination. The results obtained in both cases appear to be very promising.

First, a classification scheme using both one-dimensional texture descriptions and texture primitive descriptions was presented and classification results were discussed. The algorithm was designed to classify 12 different types of textures, including 6 random and 5 periodic textures. The random textures were taken from the Brodatz album [12]. These are grass, sand, wool, water, wood, and straw. The 5 periodic textures came from a variety of sources ranging from aerial imagery to pictures taken by the author. They are raffia, herringbone material, floor grating, aerial city, and brick wall. The algorithm achieved an overall success rate of 91.23%. The classification scheme worked well for the non-periodic texture group, and extremely well for the highly structured regular textures, achieving 100% correct classification for this group. In those cases where there was confusion additional contextual information would have been helpful. For example, knowing scale information would have aided in distinguishing wood grain from the aerial water texture. As might be expected the amount of algorithm success varied directly with the amount of structure present in the texture. Since the range of textures handled by this algorithm is varied and any confusion encountered is restricted to small groups, (2-3) of similar textures it is fair to say that the description information extracted thus far corresponds to meaning textural features.

The surface orientation determination scheme presented is in preliminary form. The method described is only partly automated. It uses schemes developed by Bajcsy [9] and Stevens [10]. It also utilizes the techniques discussed in earlier reports. Some preliminary results are presented and discussed. Tilt and slant angles are calculated for two images exhibiting non-zero texture gradients. The results for all known quantities are accurate to within 5 degrees. These results appear to be promising. However, more work needs to be done to completely automate the process.

5. REFERENCES

[1] R. Nevatia , K. Price and F. Vilnrotter, "Describing Natural Textures," USCIPI Semiannual Technical Report 860 March 1979, pp. 29-54.

[2] F. Vilnrotter, R. Nevatia and K. Price, "Automation Generation of Natural Texture Descriptions," USCIPI Semiannual Technical Report 910, September 1979, pp. 31-63.

[3] F. Vilnrotter, R. Nevatia, and K. Price, "Extraction of Texture Primitives," USCIPI Semiannual Technical Report 960, March 1980, pp. 48-59.

[4] F. Vilnrotter, R. Nevatia, and K. Price, "Determining Spatial Relationships Between Texture Primitives in Homogeneous Regular Textures," USCIPI Semiannual Technical Report 990, September 1980, pp. 10-26.

[5] F. Vilnrotter, R. Nevatia, and K. Price, "Automatic Grid Relation Extraction and Texture Reconstruction for Hmogeneous Regular Textures," USCIPI Semiannual Technical Report 1010, March 1981, pp. 27-42.

[6] R. Nevatia, K. Price, and F. Vilnrotter, "Describing Natural Textures," Proc. of the Sixth IJCAI-79, Tokyo, Japan, August 1979.

[7] F. Vilnrotter, R. Nevatia, and K. Price, "Structural Description of Natural Textures," Proc. of the Fifth IJCPR-80, Miami, Florida, December 1980.

[8] F. Vilnrotter, "Structural Analysis of Natural Textures," USC Computer Science Ph.D. Dissertation, to be published in 1981.

[9] R. Bajcsy, "Computer Identification of Textured Visual Scenes," Computer Science Report, Stanford University, 1972.

[10] K.A. Stevens, "Analysis and Representation of Visual Surface Orientation," Ph.D. Dissertation, MIT, 1978.

[11] K.A. Stevens, "Representing and Analyzing Surface Orientation," Artificial Intelligence: An MIT Perspective, pp. 104-125, Cambridge: MIT Press, 1979.

[12] P. Brodatz, Texture: A Photographic Album for Artists and Designer. New York: Dover, 1979.

[13] J.R. Kender, "Shape from Texture," Ph.D. Dissertation, Carnegie-Mellon University, 1980.

TABLE 1. CONFUSION MATRIX

Type	Floor Grating	Herring-Bone	Raffia	Brick	Aerial City	Aerial Water	Wood Grain	Straw	Grass	Sand	Wool
Floor Grating	16	0	0	0	0	0	0	0	0	0	0
Herring-Bone	0	16	0	0	0	0	0	0	0	0	0
Raffia	0	0	16	0	0	0	0	0	0	0	0
Brick	0	0	0	16	0	0	0	0	0	0	0
Aerial City	0	0	0	0	11	0	0	0	0	0	0
Aerial Water	0	0	0	0	0	15	1	0	0	0	0
Wood Grain	0	0	0	0	0	2	14	0	0	0	0
Straw	0	0	0	0	0	0	0	16	0	0	0
Grass	0	0	0	0	0	0	0	0	11	5	0
Sand	0	0	0	0	0	0	0	0	0	14	2
Wool	0	0	0	0	0	0	0	0	2	3	11

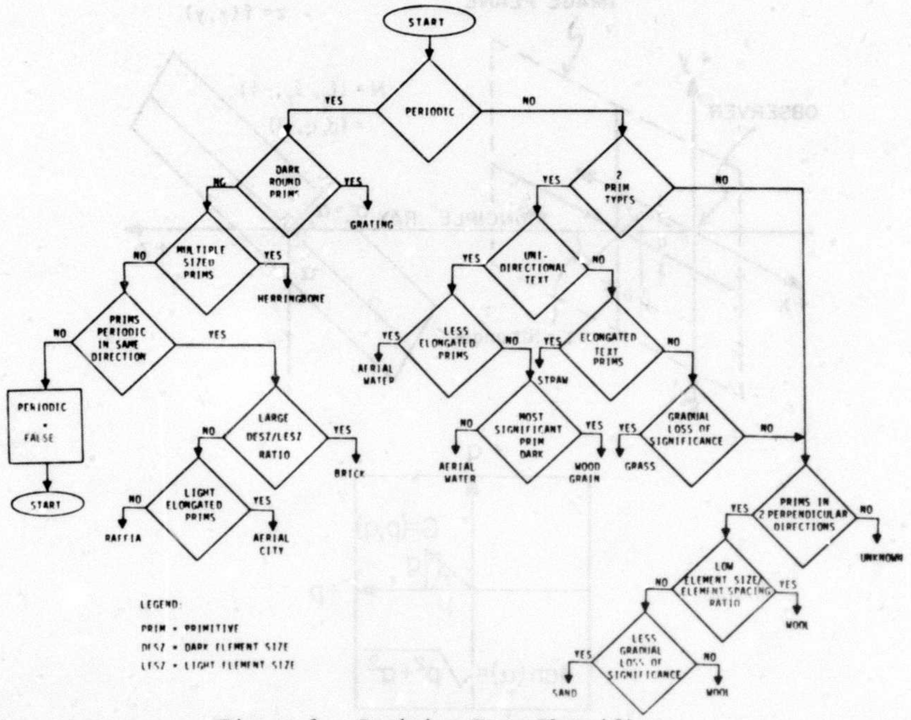


Figure 1. Decision Tree Classifier

Figure 2. Brick Wall.

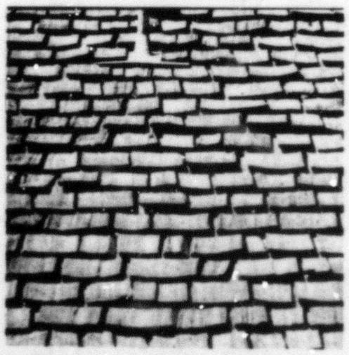


Figure 3. Redwood Shake Roof.

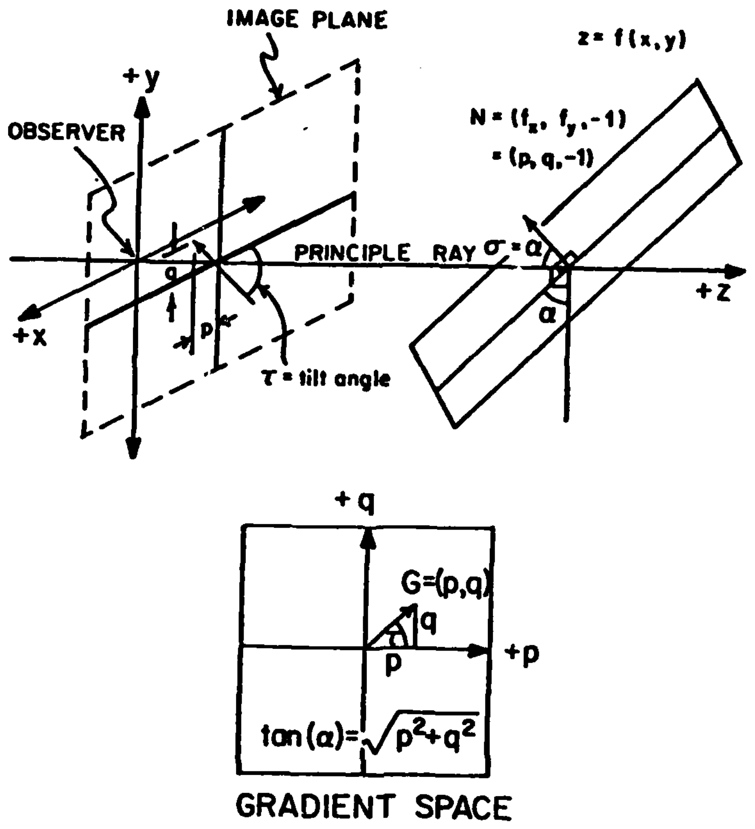


Figure 4. Gradient Space Scheme

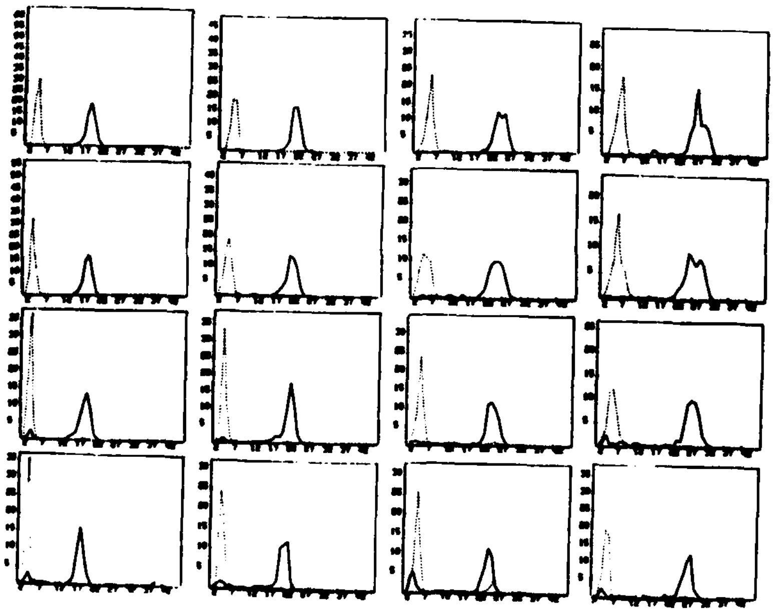


Figure 5. Vertical Brick Element Size Graphs for Example 1.

19.759	22.547	25.644	28.551
19.515	22.406	25.176	27.450
19.368	22.496	25.306	28.182
18.938	21.868	24.965	28.029

(a) Brick Vertical Dimension Matrix. Each value, d, is a dark, vertical element size.

$$S_x = -23.146 \quad S_y = -.83 \qquad (1)$$

$$S_x = 21.778 \quad S_y = -1.094 \qquad (2)$$

$$S_x = -23.566 \quad S_y = -1.864 \qquad (3)$$

$$S_x = -22.577 \quad S_y = -.381 \qquad (4)$$

(b) Gradient Results for (a).

$$\tau = \tan^{-1}\left(\frac{S_y}{S_x}\right) \qquad \sigma = \tan^{-1}\left(\frac{\sqrt{S_x^2+S_y^2}}{d}\right)$$

2.05°	2.88°
4.52°	.967°

45.95°	40.90°
46.42°	41.74°

(c) Tilt and Slant Angle Matrices.

Figure 6. Analysis for Example 1.

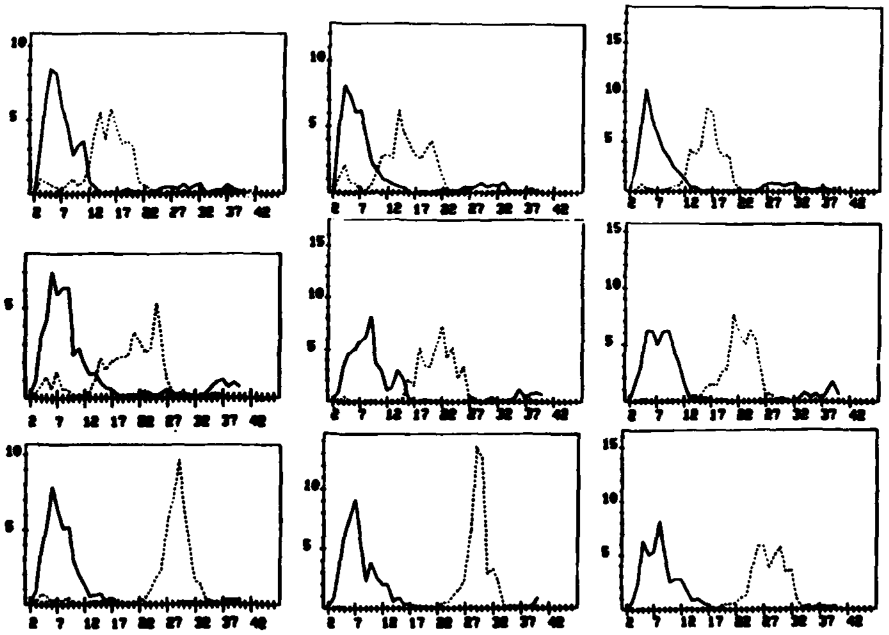


Figure 7. Vertical Brick Element Size Graphs for Example 2.

	1	2	3
d_1	16.029	15.855	16.286
d_2	21.223	21.317	22.104
d_3	28.808	29.347	28.146

Image Baseline (340 pixels)

(a) Redwood Shake Vertical Dimension Matrix. Each value, d, is a light vertical element size.

$S_x = 1.357$ $\qquad \tau = \tan^{-1}(\frac{S_y}{S_x})$

$S_y = 51.623$ $\qquad \tau = 88.49^\circ$

(b) Gradient and Tilt Angle Calculation.

$$\alpha = \tan^{-1}\left(\frac{(d_3-d_1)*\text{ Focal Distance}}{\frac{1}{2}(d_3+d_1)*\text{ Image Baseline}}\right)$$

Focal Distance = 50 mm = 2008.33 pixels

α_1	α_2	α_3
73.46°	74.17°	72.41°

(c) Slant Angle Result Matrix.

Figure 8. Analysis for Example 2

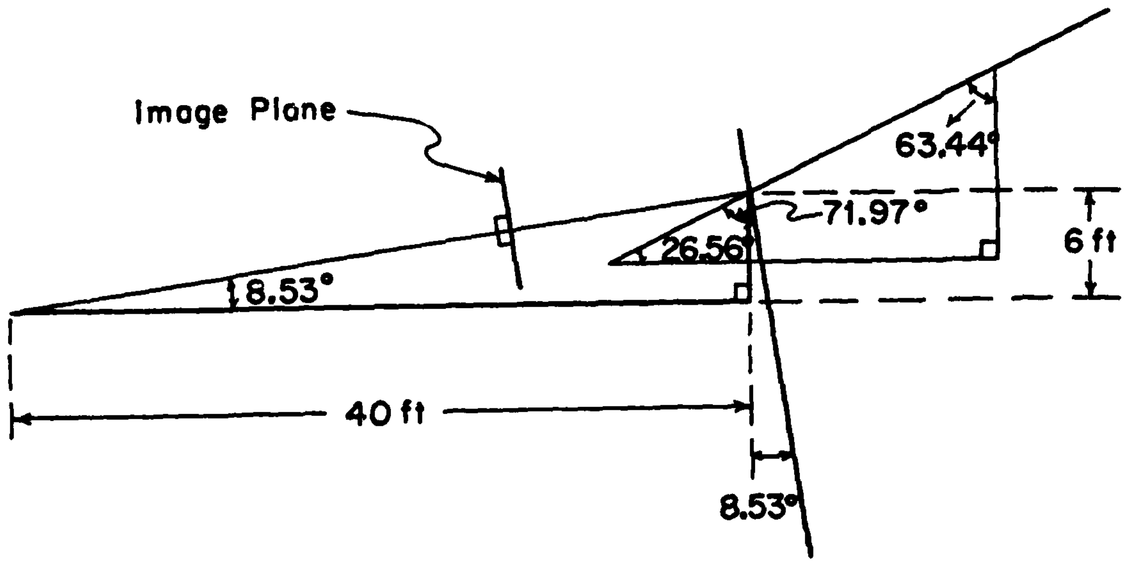


Figure 9. Geometry for Surface in Fig. 3.

AD-P000 131

DEPTH OF SCENE FROM DEPTH OF FIELD

Alex P. Pentland

SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025

ABSTRACT

Image focus is a simple function of distance to the imaged point (an effect commonly referred to as depth of field) and the parameters of the lens. Therefore, by measuring focus we can determine distance to the imaged object. Two algorithms are presented. The first determines lens-to-object distance at points of discontinuity in the scene, using one image acquired with a standard lens system. The second algorithm determines distance for all regions of the image that contain high-frequency information, using a single view acquired with a special lens system. The distance estimate produced by the second algorithm is overconstrained; thus, the accuracy of the estimate can be checked internally. These results show that single-view measurement of focus provides information comparable to measurement of stereo disparity or motion, while avoiding image-to-image-matching problems. A simple experiment is described showing that focus information is important in human perception.

1. Introduction

Most lens systems are exactly focused* at only one distance along each radius from the lens into the scene. The locus of exactly focused points forms a doubly curved, approximately spherical surface in three-dimensional space. Only when objects in the scene intersect this surface is their image exactly in focus; as the distance between the imaged point and the surface of exact focus increases, the imaged objects become progressively more defocused. This change in focus as a function of distance is known as depth of field.

The amount of defocus or blurring depends solely on the distance to the surface of exact focus and the characteristics of the lens system. Therefore, if we could measure the amount of blurring at a given point in the image, we could use our knowledge of the lens system to compute the distance to an imaged point in the scene.

The distance D to an imaged point is related to the parameters of the lens system and the amount of defocus by the following equation, which is developed in the appendix.

$$D = \frac{Fv_0}{v_0 - F - \sigma f} \tag{1}$$

where v_0 is the distance between the lens and the image plane (e.g., the film location in a camera), f the f-number of the lens system, F the focal length of the lens system, and σ the spatial constant of the imaged point's "blur circle." Image blurring is described by the point spread function, which is perhaps approximated best by a two-dimensional Gaussian $G(r,\sigma)$ with a spatial constant σ, which may be parameterized by radial distance r as follows:

$$G(r,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{2}$$

The use of a Gaussian to describe the point spread function is discussed in the appendix.

In most situations, the only unknown on the right-hand side of Equation (1) is σ, the point spread function's spatial parameter. Thus, we can use Equation (1) to solve for absolute distance given only that we can measure σ, i.e., the amount of blur at a particular image point.

I shall present two algorithms that estimate absolute distance through measurement of σ. The first algorithm uses a standard lens system, while the second uses a specially designed lens system. Both methods require only one view of the scene.

1.1 Images Acquired With A Standard Lens

Image data are determined by scene characteristics and the properties of the lens system. For example, the rate of change in image intensity at a point is dependent upon both the rate of change in scene radiance and the focus of the lens system. Therefore, if we are to measure the focus it seems that we must already know the scene characteristics — a generally unrealistic requirement.

There is, however, one special case in which we can know the scene characteristics with a considerable degree of confidence, i.e., the case when step discontinuities occur in the image formation process. Such discontinuities give rise to image data that are reliably recognizable, as will be explained below. Because the scene parameters change very rapidly at a step discontinuity, the rate of change we observe in the image is due primarily to

*"Exact focus" is taken here to mean "has the minimum variance point spread function," the phrase "measurement of focus" is taken to mean "characterise the point spread function."

the point spread function. Thus, at sharp discontinuities we can use the image data to determine the focus. These observations lead to the following algorithm for recovering the absolute distance D at points of discontinuity in an image acquired with a conventional lens.

Step (1). Find points of step discontinuity in the image formation process. We may identify such points by finding the zero crossings of $C(x,y)$, the convolution of the Laplacian of a Gaussian with the image intensity values $I(x,y)$ [1], i.e., the zero crossings of

$$C(x,y) = \nabla^2 G(r,\sigma) \otimes I(x,y)$$

which have symmetric absolute values about the zero point and which have nonzero change in image intensity across them. The following example shows that such nontrivial, symmetric zero crossings correspond closely to step discontinuities in the image formation process.

Consider a step-discontinuity edge in the image of magnitude k in the x direction at position (x_0, y_0), as defined by

$$S(x,y) = \begin{cases} k, & \text{if } x \geq x_0; \\ 0, & \text{if } x < x_0. \end{cases}$$

In this case the convolution values $C(x,y)$ have the form

$$\begin{aligned} C(x,y) &= \nabla^2 G(r,\sigma) \otimes S(x,y) \\ &= \int\int \nabla^2 G(\sqrt{(x-u)^2 + (y-v)^2}, \sigma) S(u,v) du dv \\ &= k(dG(x - x_0, \sigma)/dx) \end{aligned}$$

where $G(x - x_0, \sigma)$ is a one-dimensional Gaussian centered at point x_0, and the constant σ is the spatial constant of the point spread function at that point in the imag.. Thus, zero crossings with this functional form and with $k > 0$ occur at points of step discontinuity in the image formation process. It is unusual to find such nontrivial, symmetric zero crossings at other points in the image [2].

Step (2). Calculate the spatial constant of the point spread function. The above example shows the form of the convolution values across a step discontinuity for any given state of focus. Thus, we can use this model of the image values to measure the focus (i.e., estimate σ) at points of step discontinuity.

A maximum-likelihood estimate of σ can be formed as follows:

$$\begin{aligned} C(x,y) &= k \frac{dG(x,\sigma)}{dx} \\ &= \frac{-kx}{\sqrt{2\pi\sigma^3}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \end{aligned} \qquad (3)$$

where x, y and k are as before, and for convenience x_0 is taken to be zero. Taking the absolute value and then the natural log, we find

$$\ln \frac{k}{\sqrt{2\pi\sigma^3}} - \frac{x^2}{2\sigma^2} = \ln \left| \frac{C(x,y)}{x} \right| \qquad (4)$$

We can formulate Equation (4) as

$$Ax^2 + B = C \qquad (5)$$

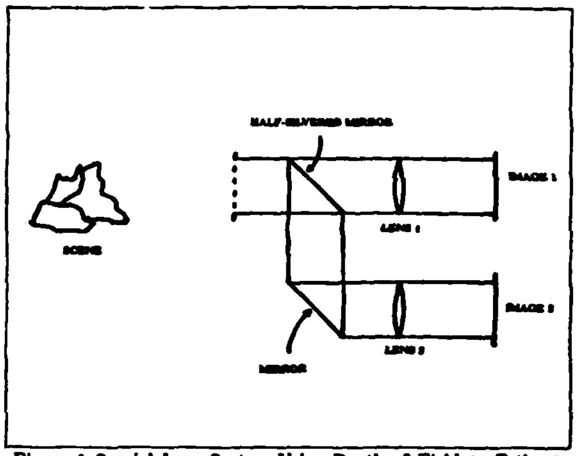


Figure 1 Special Lens System Using Depth of Field to Estimate Distance. The light from a single view is split into two identical images, using a half-silvered mirror. The images are then directed through two lens systems with different aperture size. The two resulting images are identical except for apeture, and thus objects not on the exact-focus surface have different amounts of defocus. Distance can be computed by comparing the two point spread functions at each point in the images.

where

$$A = -\frac{1}{2\sigma^2} \qquad B = \ln \frac{k}{\sqrt{2\pi\sigma^3}} \qquad C = \ln \left| \frac{C(x,y)}{x} \right|$$

If we interpret Equation (5) as a linear regression in x^2 we can then obtain a maximum-likelihood estimate of the constants A and B, and thus obtain a maximum-likelihood estimate of σ. The solution of this linear regression is

$$A = \frac{\sum_i (x_i^2 - \bar{x}^2) C_i}{\sum_i (x_i^2 - \bar{x}^2)^2} \qquad B = \bar{C} - \bar{x}^2 A \qquad (6)$$

where $\bar{x}$ is the mean of the x_i, and $\bar{C}$ is the mean of the C_i. From A we can obtain the following maximum-likelihood estimate of the value of the spatial constant σ

$$\sigma = (-2A)^{-2}$$

Step (3). Calculate the distance. Once we have obtained σ for each symmetrical, nontrivial zero-crossing point, we may employ Equation (1) to find the distance to the imaged point. An example of this algorithm applied to a natural image is shown after the next section.

1.2 Images Acquired With A Special Lens System

The limiting factor in the previous method is the requirement that we must know the scene characteristics before we can measure the focus. This requirement restricts the applicability of the method to certain special points, such as step discontinuities. If, however, we had two images of exactly the same scene, but with different depth of field, we could factor out the contribution of the scene to the two images (as the contribution is the same), and measure the focus directly.

The lens system shown in Figure 1 takes a single view of the scene and produces two images that are identical except for aperture size — and therefore depth of field. This lens system uses a half-silvered mirror (or comparable contrivance) to split the original image into two identical images, which are then directed through lens systems with different aperture size. This results in two images that are identical* except for aperture.** The difference in aperture results in differing depth of field, and thus imaged points will be focused differently in the two images.

Because the two images are identical except for aperture size they may be compared *directly*; *i.e., there is no matching* problem as there is with stereo or motion algorithms. We can then recover the absolute distance D by comparing the focus in the two images, as described in the following.

Step (1). Determine σ_1, the point spread function spatial constant for the first image, and σ_2, the point spread function spatial constant for the second image.

We start by taking a patch $f_1(r, \theta)$ of $I_1(x, y)$, the first image, centered at (x_0, y_0),

$$f_1(r, \theta) = I_1(x_0 + r\cos\theta, y_0 + r\sin\theta)$$

and calculate its two-dimensional Fourier transform $\mathcal{F}_1(\lambda, \theta)$. The same is done for a patch $f_2(r, \theta)$ at the corresponding point in the second image, giving us $\mathcal{F}_2(\lambda, \theta)$. Note that there is no matching problem, as the images are identical except for depth of field.

Now consider the relation of f_1 to f_2. Both cover the same region in the image, so that if there were no blurring both would be equal to the same intensity function $f_0(r, \theta)$. However, because there is blurring (with spatial constants σ_1 and σ_2), we have

$$\begin{aligned} f_1(r, \theta) &= f_0(r, \theta) \otimes G(r, \sigma_1) \\ f_2(r, \theta) &= f_0(r, \theta) \otimes G(r, \sigma_2) \end{aligned} \tag{7}$$

One point of caution here is that Equation (7) may be substantially in error in cases with a large amount of defocus, as points neighboring the patchs f_1, f_2 will be "spread out" into the patches by differing amounts. This problem can be avoided by using patches whose edges trail off smoothly, e.g.,

$$f_1(r, \theta) = I(x_0 + r\cos\theta, y_0 + r\sin\theta)G(r, \omega)$$

for appropriate spatial parameter ω. Noting that

$$f(r, \theta) = e^{-\pi r^2} \qquad \mathcal{F}(\lambda, \theta) = e^{-\pi\lambda^2}$$

are a Fourier pair and that if $f(r, \theta)$ and $\mathcal{F}(\lambda, \theta)$ are a Fourier pair then so are

$$f(\alpha r, \theta) \qquad \frac{1}{|\alpha|}\mathcal{F}(\frac{\lambda}{\alpha}, \theta) \quad ,$$

we see that we may use Equation (7) to derive the following relationship $\mathcal{F}_1$ and $\mathcal{F}_2$ (the Fourier transforms of image patches f_1 and f_2) and $\mathcal{F}_0$ (the transform of f_0):

$$\begin{aligned} \mathcal{F}_1(\lambda, \theta) &= \frac{\mathcal{F}_0(\lambda, \theta)G(\lambda, \frac{1}{\sqrt{2\pi}\sigma_1})}{\sqrt{2\pi}\sigma_1} \\ \mathcal{F}_2(\lambda, \theta) &= \frac{\mathcal{F}_0(\lambda, \theta)G(\lambda, \frac{1}{\sqrt{2\pi}\sigma_2})}{\sqrt{2\pi}\sigma_2} \end{aligned} \tag{8}$$

Thus*

$$\frac{\mathcal{F}_1(\lambda)}{\mathcal{F}_2(\lambda)} = \frac{G(\lambda, \sigma_1)\sigma_2}{G(\lambda, \sigma_2)\sigma_1} = \frac{\sigma_2^2}{\sigma_1^2}\exp\left(\lambda^2 2\pi^2(\sigma_2^2 - \sigma_1^2)\right) \tag{9}$$

where

$$\mathcal{F}(\lambda) = \int_{-\pi}^{\pi} \mathcal{F}(\lambda, \theta)d\theta$$

Thus, given $\mathcal{F}_1$ and $\mathcal{F}_2$ we can find σ_1 and σ_2, as follows. Taking the natural log of Equation (9) we obtain

$$\ln\frac{\sigma_2^2}{\sigma_1^2} + \lambda^2 2\pi^2(\sigma_2^2 - \sigma_1^2) = \ln\mathcal{F}_1(\lambda) - \ln\mathcal{F}_2(\lambda)$$

We may formulate this as

$$A\lambda^2 + B = C$$

where

$$A = 2\pi^2(\sigma_2^2 - \sigma_1^2) \qquad B = \ln\frac{\sigma_2^2}{\sigma_1^2} \qquad C = \ln\mathcal{F}_1(\lambda) - \ln\mathcal{F}_2(\lambda)$$

i.e., as a linear regression equation in λ^2. The solution to this regression equation is the same as shown in the last example, and gives us maximum-likelihood estimates of A and B. Solving A and B for σ_1 and σ_2 yields

$$\sigma_1 = \sqrt{\frac{A}{2\pi^2(e^B - 1)}} \qquad \sigma_2 = \sqrt{\frac{Ae^B}{2\pi^2(e^B - 1)}} \tag{10}$$

Step (2). Use the estimates of σ_1 and σ_2 from Equation (10) to calculate absolute distance to the imaged surface patch. Using Equation (1) for each of the two images, we see that we now have

$$D = \frac{Fv_0}{v_0 - F - \sigma_1 f_1} \qquad D = \frac{Fv_0}{v_0 - F - \sigma_2 f_2}$$

where f_1 and f_2 are the f-numbers for the two halves of the imaging system. We may solve either of these two equations for D, the distance to the imaged surface patch. The solution is overconstrained; *both* solutions must produce the same estimate of distance — otherwise the estimates of σ_1 and σ_2 must be in error. This can occur, for instance, when there is insufficient high-frequency information in the image patch to enable the change in focus to be calculated.

*There will be a overall brightness difference; this can be removed by multiplying the intensity measurements from one of the images by the inverse of the ratio of the mean brightnesses.

**Alternatively, one may might change the focal length instead of the aperture size. The technique, mathematics, and result remain the same.

*Note that we need o[illegible] consider the amplitude of the transforms in these calculat[illegible].

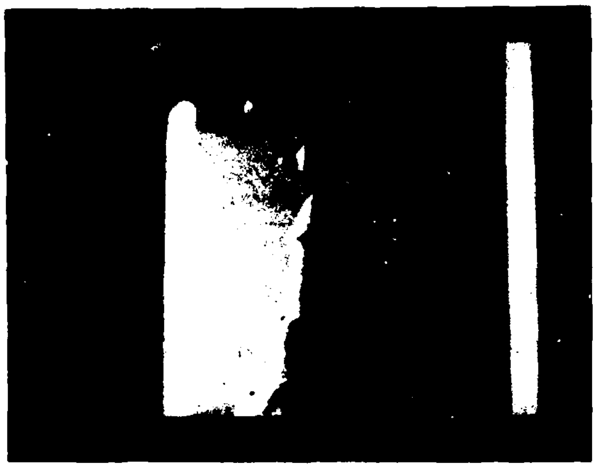


Figure 2 An Indoor Image of a Sand Castle, Refrigerator, and Door.

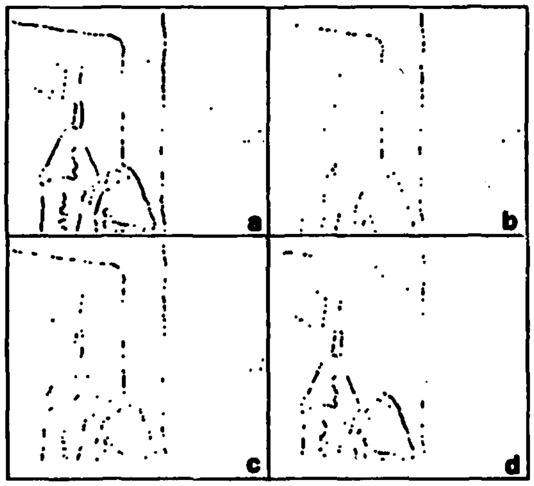


Figure 3 Point-by-Point Depth Estimates. Part (a) of this figure shows all the symmetric, nontrivial zero crossings. Parts (b), (c), and (d) show points on these zero crossings that have large, medium and small depth values, respectively. Close examination of these figures will show that the depth-from-focus algorithm is indeed working properly, although there is variability in the estimates.

2. Evaluation

The first method of deriving depth from focus, which uses a standard lens system, was implemented in a straightforward manner, and evaluated on the image shown in Figure 2. The second algorithm has not yet been evaluated, because an appropriate image pair has not yet been acquired.

Figure 3 shows the depth estimates which were obtained when the standard-lens algorithm was applied to the image of Figure 2. Part (a) of this Figure 3 shows all the symmetric, nontrivial zero crossings, i.e., identified points of step discontinuity. Parts (b), (c), and (d) show points on these zero crossings that have large, medium, and small depth values, respectively. Close examination of these figures will show that the depth-from-focus algorithm is indeed working properly, although there is variability in the estimate. This variability may have resulted from the substantial noise which was present in the digitized image values, or from using too simple an approximation to for the point spread function (this is discussed in the appendix).

One method of minimizing this variability is to average the depth values along zero-crossing segments. Such averaging, of course, makes the implicit assumption that depth values along the contour vary smoothly. To achieve useful averaging, therefore, zero-crossing contours were segmented at points of high local curvature [2], and the average depth was computed for each segment. The results of this procedure are shown in Figure 4.

Part (a) of Figure 4 shows all the symmetric, nontrivial zero-crossing segments, i.e., contour segments identified as discontinuous. Parts (b), (c), and (d) show the zero-crossing segments with large, medium and small depth values. It can be seen that the image is properly segmented with respect to depth, with the exception of one small segment near the top of (c). This mistake could be remedied by a segmentation of the zero crossings which makes allowances for positional quantization error.

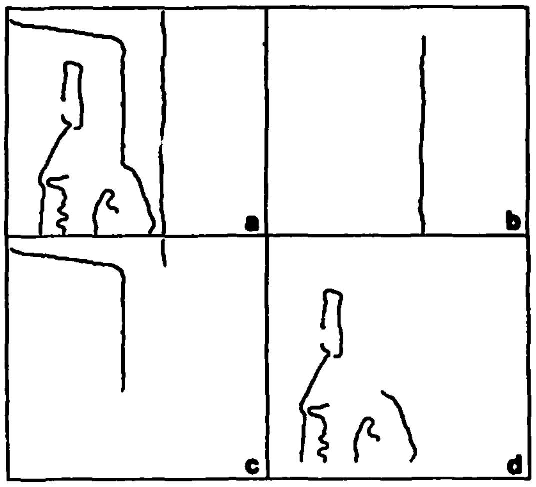


Figure 4 Depth Estimates for Zero-Crossing Segments. Part (a) of this figure shows all the symmetric, nontrivial zero-crossing segments. Parts (b), (c), and (d) show the zero-crossing segments that have large, medium and small depth values, respectively. It can be seen that the image is properly segmented with respect to depth, with the exception of one small segment near the top of (c). This mistake could be remedied by a segmentation procedure which is tolerent of positional quantization errors of the location of the zero crossings.

3. Discussion

The most striking property possessed by these two algorithms is that absolute depth can be recovered from a single view with no image-to-image matching, the major problem in stereo and motion algorithms. Furthermore, no special scene characteristics need be assumed, so that the techniques are generally applicable.

The normal-lens algorithm appears to have a potential for accurate depth estimates that is comparable to edge- or feature-based stereo and motion algorithms (e.g., [3], [4] or [5]). Each of these algorithms is able to recover scene depth at certain feature points given a camera model. The major difficulty in using stereo and motion algorithms is identification of the same point in successive views, whereas the major difficulty in measuring focus at discontinuities is identification of the discontinuities; once this has been accomplished, the rest is straightforward. We have suggested one technique for identifying discontinuities, but the possibility of better methods is not excluded.

The special-lens algorithm provides considerably stronger information about the scene because it *overconstrains* scene depth, allowing an internal check on the algorithm's answer. Thus, the special-lens algorithm provides information comparable to the best that is theoretically available from three-or-more-image stereo and motion algorithms. The major limitation of the special-lens algorithm appears to be that it requires sufficient high-frequency information to measure the change in focus. This is roughly similar to the requirement of stereo and motion algorithms that there be distinguishable image features.

One question concerning the use of focus information is whether such information is sufficiently free of noise to be useful. Research in estimating the point spread function of an unfamiliar image has shown that both the amplitude and phase components of the Fourier transform are sufficiently stable to allow useful estimation of the point spread function in the presence of normal image noise [6, 7] Thus, it appears that the issue of noise is not likely to be an insurmountable hinderence.

Perhaps the major issue in the practical application of focus-measuring algorithms is resolution. The normal-lens algorithm can be applied to virtually any image; however, it requires that the digitization be fine enough to adequately resolve the point spread function. For a 35-mm slide taken in the normal manner, this may mean digitizing with the 12 micron resolution available on better-quality digitizers. The resulting image, of course, will have somewhat more pixels than is currently the norm. This plethora of pixels can be averted by using a combination of focal length and f-number that results in a relatively smaller depth of field than is typically employed by most photographers, as was done in the example shown here. It is worth noting that the human eye typically has a f-number of approximately 4, and thus has exactly such limited depth of field.

Resolution is also an issue for the special-lens algorithm, as the resolution of the resulting depth map is the original image resolution divided by the size of the region used to calculate the Fourier transform. It appears that a region of perhaps 16×16 pixels should be used in obtaining the transform, and thus the resolution of the depth map will be $1/16^{th}$ the resolution of the original image.

Human Perception. It is interesting to note that the human visual system measures the information needed by the special-lens system in at least two ways. First, the depth of field for the red-green retinal cells is different from that for the blue retinal cells, because of one diopter of chromatic aberration in the lens. This provides two simultaneous views of the scene with dissimilar depth of field, albeit in different spectral bands. Second, the focal length of the human eye is constantly varying in a sinusoidal fashion at a frequency of about 2 hz [8]. The range of variation depends upon the average accommodation* [9], but can be almost one diopter** under normal conditions. Thus, two views of the same scene with different depth of field are obtained within two hundred and fifty milliseconds, approximately the duration of a typical fixation.

A experiment demonstrating the importance of depth of field in human perception can be easily performed by the reader. Make a pinhole camera by poking a hole through a piece of paper with a ball-point pen. Imposition of a pinhole in the line of sight causes the depth of field to be very large, thus effectively removing this depth cue from the image. Close one eye and view the world through the pinhole, noting your impression of depth. Now quickly remove the pinhole and view the world normally (still using only one eye). The change in the sense of depth caused by adding or removing the depth-of-field distance cue is remarkable; observers report that the change is fully comparable to the change caused by going from monocular viewing to binocular viewing, or the change which occurs when a stationary object begins to move.

The effect of the pinhole is not due to change in the field of view, as can be demonstrated by comparing the percept obtained through the pinhole to the percept obtained through a viewing tube which occludes a similar portion of the scene. The effect is also not due to removal of the accomodation depth cue because, although it is true that pinhole viewing makes the eye's accommodative state irrelevant, accomodation can provide an estimate of depth at only a single point [10]. Thus removing the accommodation cue can only change one's impression of the average distance or of distance to the central point, not of the depth relations throughout the scene.

*Accomodation refers to the focal length of the eye's lens.

**Diopters are the reciprocal of focal length (i.e., $D = 1/F$). One diopter is approximately the strength of one's first pair of glasses.

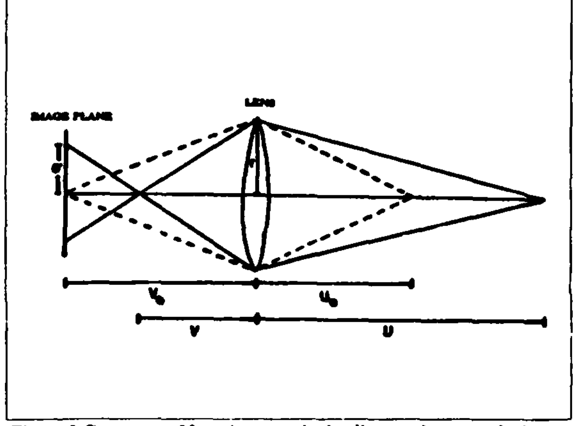


Figure 5 Geometry of Imaging. v_0 is the distance between the image plane and the lens, u_0 is the distance between the lens and the locus of perfect focus, and r is the radius of the lens. When an point at distance $u > u_0$ is projected through the lens, it focuses at a distance $v < v_0$, so that a blur circle is formed.

4. Appendix

For a thin lens,

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{F} \tag{11}$$

where u is the distance between a point in the scene and the lens, v the distance between the lens and the plane on which the image is in perfect focus, and F the focal length of the lens. Thus,

$$u = \frac{Fv}{v - F} \tag{12}$$

For a particular lens, F is a constant. If we then fix the distance v between the lens and the image plane to the value $v = v_0$, we have also determined a locus of points at distance $u = u_0$ that will be in perfect focus, i.e.,

$$u_0 = \frac{Fv_0}{v_0 - F} \tag{13}$$

We may now explore what happens when a point at a distance $u > u_0$ is imaged. Figure 5 shows the situation in which a lens of radius r is used to project a point at distance u onto an image plane at distance v_0 behind the lens. Given this configuration, the point would be focused at distance v behind the lens — but in front of the image plane. Thus, a blur circle is formed on the image plane. Note that a point at distance $u < u_0$ also forms a blur circle; throughout this paper we assume that the lens system is focused on the nearest point so that u is always greater than u_0. This restriction in not neccessary in the second algorithm, as overconstraint on the distance solution allows determination of whether $D = u > u_0$ or $D = u < u_0$.

From the geometry of Figure 5 we see that

$$\tan\theta = \frac{r}{v} = \frac{\sigma}{v_0 - v} \tag{14}$$

Combining Equations (12) and (14) and substituting the distance D for the variable u we obtain

$$D = \frac{Frv_0}{rv_0 - F(r + \sigma)}$$

or

$$D = \frac{Fv_0}{v_0 - F - \sigma f}$$

where f is the f-number of the lens.

The blurring of the image is better described by the point spread function than by a blur circle, although the blurring is bounded by the blur circle radius in the sense that the point spread function is less than some threshold outside of the blur circle. The point spread function is due primarily to diffraction effects, which for any particular wavelength produce wave cancellation and reinforcement resulting in intensity patterns qualitatively similar to the sinc function, $\frac{\sin r}{r}$, but with different amplitudes and periods for the "rings" around the central peak.

The point spread function describes the image intensity $I(\mu, \nu)$ caused by a single coherent point-source light in terms of the parameters of the lens system. It is described [11] by

$$I(\mu, \nu) = \left(\frac{k\sigma^2}{2F^2\mu}\right)^2 (U_1^2(\mu, \nu) + U_2^2(\mu, \nu))$$

where

$$U_m(\mu, \nu) = \sum_{s=0}^{\infty} (-1)^s \left(\frac{\mu}{\nu}\right)^{m+2s} J_{m+2s}(\nu)$$
$$\mu = k\left(\frac{\sigma}{F}\right)^2 (v - v_0)$$
$$\nu = k\left(\frac{\sigma}{F}\right) r$$
$$k = \frac{2\pi}{\lambda}$$

where λ is the wavelength of the light, r is the distance from the center of the point spread function, $J_n(\nu)$ is the Bessel function of the first kind and order n, and v_0, v, σ and F are as before.

The "rings" produced by this function vary in amplitude, width and position with different states of focus and with different wavelengths. As wavelength varies these rings change position by as much as 90°, so that the blue light troughs become positioned over the red light peaks, etc. Further, change in wavelength results in substantial changes in the amplitude of the various rings. Although this point spread function is quite complex, and the sum over different wavelengths even more so, it appears that the envelope for the various functions has the general shape of a two-dimensional Gaussian.

Sampling effects caused by digitization are typically next in importance after the diffraction effects. The effect of sampling may be accounted for in the point spread function by convolving the above diffraction-produced point spread function with functions of the form $\frac{\sin r}{r}$. Other factors such as chromatic abberation, movement, and diffusion of photographic emulsion may also be accounted for in the final point spread function by additional convolutions.

The net effect, in light of the central limit theorem, is probably best described by a two-dimensional Gaussian $G(r, \sigma)$ with some spatial constant σ, although the question is still being investigated.* The spatial constant σ of the point spread function will be proportional to the radius of the blur circle; however the constant of proportionality will depend on the particulars of the optics, sampling, etc. In this paper the radius of the blur circle and the spatial constant of the point spread function have been treated as identical; in practical application where recovery of absolute distance is desired the constant of proportionality k must be determined for the system and included in Equation (1) as follows:

$$D = \frac{Fv_0}{v_0 - F - \sigma k f}$$

REFERENCES

[1] E. Hildreth, Implementation of a Theory of Edge Detection,*M.I.T. AI Laboratory Technical Report 579* (April 1980).

[2] A. Pentland, The Visual Inference of Shape: Computation from Local Features,*Ph.D. thesis, Department of Psychology, Massachusetts Institute of Technology* (1982).

[3] D. Marr and T. Poggio, A Computational Theory of Human Stereo Vision,*Proc. R. Soc. Lond.* **204**, B (1979), 301-328.

[4] H. H. Baker and T.O. Binford, Depth from Edge and Intensity Based Stereo,*Proc. 7th IJCAI, Vancouver, B.C.* (1981), 631-636.

[5] S. Ullman, *The Interpretation of visual motion*, M.I.T. Press, Cambridge, Mass., 1979.

[6] K. T. Knox and B. J. Thomson, Recovery of Images from Atmospherically Degraded Short-Exposure Photographs,*Astrophys. J.* **193**, L45-L48 (1974).

[7] J. B. Morton and H. C. Andrews, A Posteriori Method of Image Restoration,*Opt. Soc. Am.* **69**, 2 (1979), 280-290.

[8] A. Arnuff and O. Dupuy, Contribution à l'Etude des Microfluctuations d'Accommodation de l'Oeil,*Rev Opt* **39**, 5 (1960), 195-208.

[9] F. W. Cambell, J. G. Robson, and G. Westheimer, Fluctuations of Accommodation Under Steady Viewing Conditions,*J. Physiol* **145** (1959), 579-594.

[10] H. Crane, *A Theoretical Analysis of the Visual Accommodation System in Humans,* Final Report NAS 2-2760, NASA Ames Research Center, 1966.

[11] M. Born and E. Wolf, *Principles of Optics,* Pergamon, London, 1965.

*Even if effects other than diffraction are very small, the resulting point spread function can be quite well modeled by the difference of three Gaussians. Using this model of the point spread function makes the computations of Equations (7) - (10) somewhat more difficult, but the nature of the result is unaffected.

AD-P000 132

Reasoning between Structure and Function

Michael R. Lowry

Stanford Artificial Intelligence Laboratory
Stanford University, Stanford, CA 94305 USA

Abstract

This report describes conceptual work on a system that reasons between structure and function in the domain of physical objects interacting through mechanical contact. Structure is represented by a hierarchy consisting of generalized cones, abstract shape descriptions, and motion sequences of the spines of generalized cones. Function is represented by a hierarchy of kinematic primitives, functional primitives, and causal networks. Both qualitative and quantitative reasoning are used to determine the relationship between structure and function.

Overview

The objective of high level vision research has been to recover the 3-d structure of a scene from image data. In order for an intelligent entity to carry out tasks in the real world the perceived 3-D structure of a scene is transformed into objects indentified by functional category. Comparison to a set of structural prototypes is one method to do this transformation. However it is inherently weak when confronted with objects differing from the stored prototypes since there is no good object independent similarity metric. It is also unable to use objects adaptively- e.g. to see that a dime can be used as a screwdriver.

A system that understands and reasons between structure and function would have many applications. This ability is a necessary component of an intelligent robot, able to carry out tasks in unconstrained enviorments. The system could serve as an intelligent assistant for mechanical designers. This system could also solve the crucial difficulty in the parts inspection problem of classifying structural anomalies as cosmetic flaws or functional impairments. Since humans understand the world in functional terms, it would facilitate human interface.

The problem of reasoning between structure and function is generic across domains. AI researchers are developing automatic programming systems which take functional specifications and produce computer programs, while others have focused on digital systems, analog circuits, and organic chemistry. Previous work in the domain of objects interacting through mechanical contact have either used geometric domains simplified to 1 or 2 dimensions with a limited vocabulary of physical interaction [Forbus 81a] [deKleer 75], or have focused on the causal interaction in complex mechanisms [Rieger 78]. This report focuses on the relation between 3-d geometry and function.

The first section of the paper discusses the representation of function for 3-d objects interacting through mechanical contact. Although it is not exhaustive, it shows how functional properties can be built up out of physical properties associated with structural properties. The second section discusses generalized cones as an appropriate representation of 3 dimensional objects for determining kinematic properties. The third section discusses representations and methods for reasoning between structure and function, including some preliminary work on the use of analogy.

Functional Representation

The function of an object is represented as a hierarchical description with symbolic causal relationships at the top level and kinematic primitives at the bottom level. At an intermediate level are functional primitives, which are used as nodes in the causal relationships and are instantiated by the kinematic primitives. The following are some examples of functional primitives, represented as triplets of the form (function agent object):

(grasp hand screwdriver)
(support table lamp)
(contain glass water)
(support chair human being)
(cut knife meat)
(contain closet clothes)
(brace strut chair-leg)

The function of an object is the causal relationships in which an object is a casual agent. Causality is treated as a simplifying mental construct for the purposes of description and reasoning rather than a metaphysical property. Thus it is possible for events linked by a causal relationship to be reversed in the normal time sequence [deKleer 79]. The following is an example of a causal network of a set of functional primitive triplets:

(grasp robot-hand screwdriver) and (rotate robot-hand) cause (rotate screwdriver)

(mated screwdriver-tip screw-slot) and (rotate screwdriver) cause (rotate screw)

The functional primitives are instantiated as kinematic primitives. The kinematic primitives are motions, constraints on motions, and the relationship of these two to various types of forces. For example the functional primitive (support table lamp) is instantiated as a constraint on motion in the direction of gravity imposed by the table on the lamp. The force associated with this constraint is that necessary to either bend the table or to break the table. Also associated with the interaction between the table and the lamp are frictional forces that constrain the motion of the lamp in the plane of the table's surface. These forces could be represented as numerical quantities, algebraic expressions, or more abstractly as a partial order. For example th bending force is less than the breaking force, while the rolling coefficient of friction is less than the sliding coefficient of friction. Ken Forbus' work on quantity space shows how a partial order of quantities is useful for qualitative reasoning [Forbus 81b]. The description of motion and forces requires a co-ordinate system for the associated vectors. The next section discusses how the choice of co-ordinate systems constrains the choice of a representation for structure.

Structural Representation

For objects interacting through mechanical contact, the key components of the structure are the volume, surfaces, edges, and points and the mechanical forces associated with these geometric properties. Objects occupy a volume, and some amount of force is needed to deform this volume or fragment the object. Surfaces have static, sliding, and rolling coefficients of friction. A sharp edge can cut, while a point can pierce. The co-ordinate system for motions and forces should be centered in the objects. For example, consider a description of a man walking first in the co-ordinate systems of the body and limbs of the man and then in the co-ordinate system of a camera. The description of motion is simple and concise in the first set of co-ordinate systems, but in the camera-centered co-ordinate system the motion of the legs, arms, and hands are very complex. [Marr 76] argues for an object-centered co-ordinate system with one axis along the direction of elongation. The choice is dictated by the domain, for example in the domain of free-falling objects in a gravitational field a global co-ordinate system oriented along the direction of gravity would facilitate the description of motion. Generalized cones [Binford 71] are a concise representation of 3-D objects with an object-centered co-ordinate system oriented along an axis of elongation. [Nevatia 74] describes a program that builds a generalized cone description of a scene from a laser-rangefinder depth map.

As implemented in the Acronym system generalized cones provide a simple constructive representation of shape[Brooks 81]. Volumes are 2-D cross-sections swept along a spine while surfaces are either the end-faces or the perimeter of the cross-section swept along the spine. Similar properties hold for edges and points. This description facilitates the transformation of surface, edge, and point co-ordinate systems into the co-ordinate system of the body of the object. Thus the forces arising between surfaces of contacting objects expressed in surface co-ordinate systems can be easily transformed into motions and constraints on motion of the objects themselves. The constructive analytic representation of generalized cones in Acronym also facilitates the computation and reasoning about mechanical properties such as weight, surface area, and frontal cross-section for computing drag.

While generalized cones and the mechanical properties of objects provide a good representation for reasoning between structure and kinematic primitives, more abstract representations of structure facilitate reasoning between structure and functional primitives. The distinction between accidental and criterial structural features is often brought out through more abstract descriptions. For example structural symmetry is a good indication of functional symmetry. Grouping of objects is often associated with a particular function, for example grouped objects that are physically cross-linked are often some type of support structure. Protrusions and intrusions are used when two objects are mated together for some purpose, such as the tip of a screwdriver mating the slot in the head of a screw. Negative volumes open at one end suggest containment of solids or liquids. [Hollerbach 75] discusses computing descriptions of intrusions and protrusions from a sub-class of generalized cones.

At the most abstract level, structure could be described topologically or through the spines of generalized cones. Topological representations of structure, such as connected, adjacent, and contain are rather weak except for domains which are inherently topological such as graphs. An example would be a network of roads. The spines of generalized cones correspond to stick figure diagrams, which lead to the strong impression of causal interactions when their motion through time is simulated.[Soroka 80] describes using a simulation of the time sequence of positions of the spines of a robot arm to debug robot assembly programs. [Marr 80] discusses using a state-motion-state description to segment actions based on information in the relative positions of the spines of generalized cones through time.

Relationship between Structure and Function

The relationship between structure and function depends on the level of description. At the level of kinematic primitives and generalized cones the relationship can be described using parametric constraints on shapes and mechanical properties. For example, a lamp will stay in one fixed spot on the plane of a table as long as lateral forces do not exceed the static frictional forces between the surface of the table and the bottom of the lamp. A robot hand rigidly grasps a screwdriver if the forces applied by the hand constraining the six degrees of freedom of the screwdriver exceed any other forces applied. An object provides a comfortable sitting support for a human being if its dimensions match those determined by ranges of human physical dimensions. These constraints are generally inequalities over variables and terms composed of sums,products,divisions, and trigonometric functions of terms. Acronym currently has a sub-system that represents and manipulates these types of constraints.

```
Topological realationships                      Functional relationships
 and spines of generalized  --------------       represented through a
  cones indexed by time        ANALOGY              causal network
           |                                              |
           |                                              |
           |                                              |
           |                                              |
           |                                              |
 Abstract static representation
    of shape such as --------------------        Functional primitives
    symmetry, groupings,     ANALOGY                      |
     protrusions                                          |
           |                                              |
           |                                              |
    Generalized Cones ----------------------    Kinematic primitives
                         CONSTRAINTS
```

Hierarchy of functional and structural descriptions with methods of reasoning between them.

At the level of functional primitives and abstract shape descriptions the relationship can be described using enabling links and inhibiting links in the causal network. For example:

(open-concavity glass) ENABLES (contain glass liquid)

(protrusion tip-of-screwdriver) and (intrusion slot-of-screwhead) and (same-cross-section tip-of-screwdriver slot-of-screwhead) ENABLES (mated tip-of-screwdriver slot-of-screwhead)

(NOT-ALIGNED screwdriver screw) INHIBITS (mated tip-of-screwdriver slot-of-screwhead)

Work performed at MIT demonstrated that analogy can be a useful tool for reasoning between *structure* and *function* at this *level* of description[Binford 82]. Winston's analogy program coupled with Katz's natural language parser were used to input natural language descriptions of mechanical interactions such as a screwdriver turning a screw. The shape properties that enabled these interactions were also described. Using analogy the program decided that an allen wrench might plausibly be used to turn an allen bolt based on their abstract shape descriptions.

At the most abstract level of description patterns of movements between objects correspond to causal networks. For example, the structural description of the sequence of a robot hand moving to an object, moving its fingertips so that they hold the object, moving the hand to a new location, and then moving its fingertips apart has a corresponding functional description of grasping, moving, and ungrasping an object. The overall network can be represented as moving the object.

Summary

This report described conceptual work on a system that reasons between structure and function for 3-D objects interacting through mechanical contact. The hierarchical description of structure and function, and the relationship and the methods used to reason between them are shown in the preceding figure.

This work was supported by ARPA contract N00039-82-C-0250

References

[Binford 71] Binford, Thomas O., **Visual Perception by Computer,** IEEE Systems Science and Cybernetics Conference, Miami, December 1971.

[Binford 82] Binford, Thomas O., Boris Katz, Michael R. Lowry, and Patrick H. Winston, Forthcoming MIT-AI Memo 1982

[Brooks 81] Brooks, Rodney A., **Symbolic Reasoning Among 3-D Models and 2-D Images** PhD Thesis Stanford June 1981.

[deKleer 75] deKleer, Johan, **Qualitative and Quantitative Knowledge in Classical Mechanics,** MIT AI-TR-352, Cambridge, MASS December 1975.

[deKleer 79] deKleer, Johan **Causal and Teleologiacl Reasoning in Circuit Recognition,** MIT AI-TR-529, Cambridge, MASS September 1979.

[Forbus 81a] Forbus, Kenneth D., **A Study of Qualitative and Geometric Knowledge in Reasoning Avout Motion,** MIT AI-TR-615, Cambridge, MASS Feburary 1981.

[Forbus 81b] Forbus, Kenneth D., **Proposal for a Study of Commonsense Physical Reasoning,** DRAFT, Cambridge, MASS May 1981.

[Hollerbach 75] Hollerbach, J.M., **Hierarchical Shape Description of Objects by Selection and Modification of Prototypes,** *MIT-AI-TR 346, 1975*

[Marr 76] Marr, D. and H.K. Nishihara, **Representation and Recognition of the Spatial Organization of Three Dimensional Shapes,** MIT AIM-377, Cambridge, MASS August 1976.

[Marr 80] Marr, D. and Lucia Vaina, **Representation and Recognition of the Movements of Shapes,** MIT AIM-597, Cambridge, MASS October 1980.

[Nevatia 74] Nevatia, R., **Structured Descriptions of Complex Curved Objects for Recognition and Visual Memory,** Stanford AIM-250, October 1974.

[Popplestone 79] Popplestone, R.J., A.P. Ambler, and I.M.Bellos, **An Interpreter for a Language for Describing Assemblies,** DAI Research Paper No. 125, Edinburg, 1979.

Soroka 80] Soroka, Barry I., **Debugging Manipulator Programs with a Simulator,** Autofact West Conference, Society of Manufacturing Engineers, Anaheim, November 1980.

Vaina 82] Vaina, L., **From Shapes and Movements to Objects and Actions,** Synthesis 1982

AD-P000 133

Gradient Space Under Orthography and Perspective

Steven A. Shafer[1], Takeo Kanade[1], and John R. Kender[2]

[1]Computer Science Department, Carnegie-Mellon University, Pittsburgh PA
[2]Computer Science Department, Columbia University, New York NY

Abstract

Mackworth's gradient space has proven to be a useful tool for image understanding. However, descriptions of its important properties have been somewhat scattered in the literature.

This paper develops and summarizes the fundamental properties of the gradient space under orthography and perspective, and for curved surfaces. While largely a recounting of previously published results, there are a number of new observations, particularly concerning the gradient space and perspective projection. In addition, the definition and use of vector gradients as well as surface gradients provides concise notation for several results.

The properties explored in the paper include the orthographic and perspective projections themselves; the definition of gradients; the gradient space consequences of vectors (edges) belonging to one or more surfaces, and of several vectors being contained on a single surface; and the relationships between vanishing points, vanishing lines, and the gradient space.

The paper is intended as a study guide for learning about the gradient space, as well as a reference for researchers working with gradient space.

Introduction

The *gradient space* has proven a useful tool for image understanding. Since its proposal by Mackworth [12] based on Huffman's *dual space* [5], the gradient space has been used for defining consistency of line-labelings [6, 7], relating surface orientation to image intensity [3, 4, 16], and relating surface orientation to image geometry [8, 9, 10, 13, 15].

The descriptions of important gradient space properties, however, have been scattered throughout the literature. In this paper, the gradient space is defined and its fundamental properties are summarized. This presentation is especially useful because its assignment of gradients to *vectors* as well as surfaces allows concise statements of important properties.

This paper is primarily a summary and re-statement of important gradient space properties, but also includes statements of some new properties. It is intended that the paper provide a reference for people working with the gradient space, as well as a study guide for researchers being introduced to the gradient space.

Preliminary Definitions

Coordinate System

In these equations, the coordinate system being used is Mackworth's [12]: the x and y axes in the scene are aligned in the image (x horizontal, y vertical), and the z axis points *towards* the viewer (i.e. a right-handed coordinate system) (figure 1). The *eye* (center of lens) is at the origin (0,0,0), and the *image plane* is $z = -1$ (i.e. the focal plane is $z = 1$, which is rotated around the origin to the image plane, $z = -1$, to preserve the sense of "up", "down", "left", and "right" from the scene).

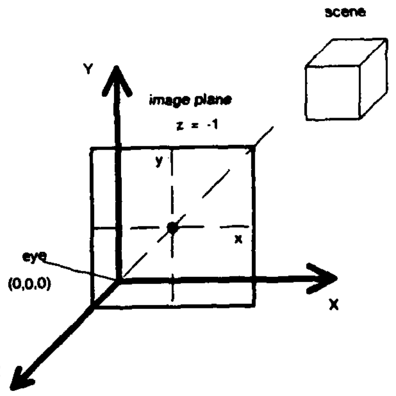


Figure 1: Coordinate System

Orthography

In *orthographic projection*, the scene point (x, y, z) is mapped onto the image point (x, y). Thus, the image point (x, y) represents the set of scene points (x, y, z) for all values of z.

Perspective

In *perspective projection*, the scene point (x, y, z) is mapped onto the image point $(-x/z, -y/z)$. The image point is the point at which a line through the origin (eye) and (x, y, z) intersects the image plane. The unit of measure in the coordinate system is the focal length of the camera lens. This is similar to Kender's coordinate system, but with the direction of the z-axis reversed [11]. An image point (x, y) *corresponds to the set of scene points $(ax, ay, -a)$ for all values of a.*

Gradient Space and Orthography

In this section important relationships between surfaces, vectors, and gradients are described. In addition, several important observations concerning *orthographic projection* are noted.

1. Definition of Surface Gradient

   Suppose a surface is defined as $-z = f(x, y)$. Then its gradient (p, q) is defined by [12]:

$$(p,q) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = \left(\frac{-\partial z}{\partial x}, \frac{-\partial z}{\partial y}\right)$$

   The set of all gradients (p, q) is the *gradient space*.

   Corollary to this result is:

   1. In any direction u, the tangent vector to the surface is:
   $\left(\frac{dx}{du}, \frac{dy}{du}, -p\frac{dx}{du} - q\frac{dy}{du}\right)$
   The tangent vectors in the directions $u = x$ and $u = y$ are thus $(1, 0, -p)$ and $(0, 1, -q)$ respectively. Their cross product, $(p, q, 1)$, is therefore a surface normal.

2. Gradient of a Plane

   Suppose a plane is defined by $Ax + By + Cz + D = 0$. Then its gradient is:

$$(p,q) = \left(\frac{A}{C}, \frac{B}{C}\right)$$

   Corollary to this result is:

   1. Since D has no effect on p and q, parallel planes have the same gradient. Each point (p, q) thus represents the gradient for a family of parallel planes.

3. Gradient of a Vector

   Suppose a vector is $(\Delta x, \Delta y, \Delta z)$. Then its gradient can be defined as:

$$(p,q) = \left(\frac{\Delta x}{\Delta z}, \frac{\Delta y}{\Delta z}\right)$$

   (This is not Huffman's *dual line* [5]; the dual is the line described below in section 7.)

   Although the term *gradient* technically refers to a property of differentiable surfaces, it is used here for vectors because the gradient space can represent 3D orientation in general, not just orientation of surfaces.

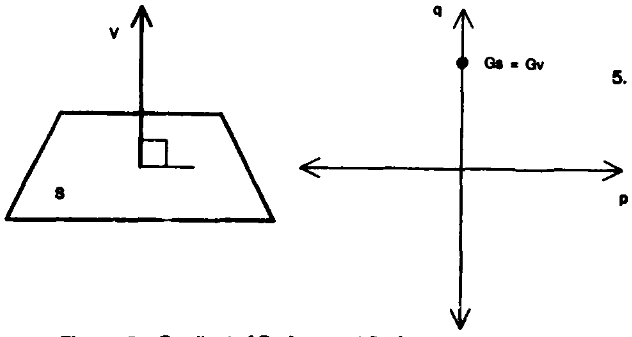


Figure 2: Gradient of Surface and Surface Normal

Corollaries of this result are:

1. Parallel vectors have the same gradient. The gradient of a line can be defined as the gradient of any vector contained in the line.

2. The gradient of a surface is the same as the gradient of its surface normal vectors (figure 2).

3. Under orthography, the vector $(\Delta x, \Delta y, \Delta z)$ is seen in the image as an edge $E = (\Delta x, \Delta y)$. If the vector's gradient is G, then $G = E / \Delta z$. The line in gradient space from the origin through G is thus parallel to the edge in the image (figure 3).

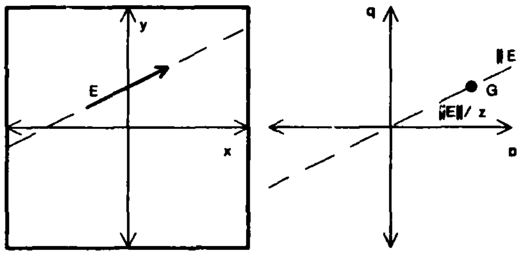


Figure 3: Gradient of a Vector Under Orthography

4. Vector Contained on a Surface

   Suppose the vector $(\Delta x, \Delta y, \Delta z)$ is contained on a surface whose gradient is (p, q).

   Since the surface normal $(p, q, 1)$ must be orthogonal to the vector,

$$(p, q, 1) \cdot (\Delta x, \Delta y, \Delta z) = 0$$
$$p\Delta x + q\Delta y + \Delta z = 0$$

   Therefore,

$$(p, q) \cdot (\Delta x, \Delta y) = -\Delta z$$

   Corollary to this result is:

   1. Under orthography, the vector $(\Delta x, \Delta y, \Delta z)$ is seen in the image as the edge $E = (\Delta x, \Delta y)$. If it is contained on a surface with gradient G, then

$$G \cdot E = -\Delta z$$

   This is one of the most important relations in orthography, since polyhedral scenes contain surfaces bounded by many edges.

5. Vector Contained on Two Surfaces

   Suppose the vector $(\Delta x, \Delta y, \Delta z)$ is the boundary between two surfaces with gradients $G_1 = (p_1, q_1)$ and $G_2 = (p_2, q_2)$ (figure 4). Further, define E to be $(\Delta x, \Delta y)$.
   Then:

$$-\Delta z = G_1 \cdot E = G_2 \cdot E$$
$$0 = (G_1 - G_2) \cdot E$$
$$(G_1 - G_2) \perp E$$

   $G_1 - G_2$ is the vector from G_1 to G_2 in the gradient space. Thus, the vector E is perpendicular to the line containing G_1 and G_2 in gradient space.

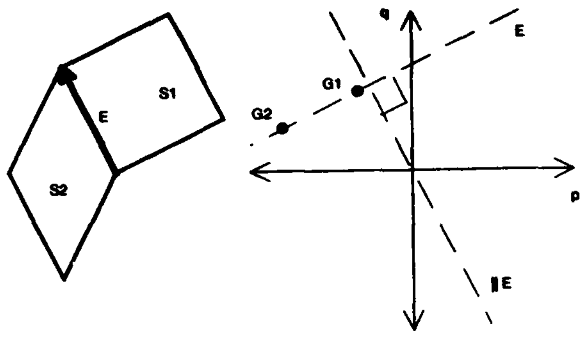


Figure 4: Vector Contained on Two Surfaces

As a corollary:

1. Under orthography, with the above definitions, the edge in the image is E. Therefore, the edge in the image is perpendicular to the line containing G_1 and G_2. (This is Mackworth's relation involving *connect edges* [12].)

6. Two Vectors Contained on a Surface
Suppose the two vectors $(\Delta x_1, \Delta y_1, \Delta z_1)$ and $(\Delta x_2, \Delta y_2, \Delta z_2)$ both lie on a surface whose gradient is $G = (p, q)$. Further, let $E_1 = (\Delta x_1, \Delta y_1)$ and $E_2 = (\Delta x_2, \Delta y_2)$ correspond to the x-y components of the two vectors (figure 5).

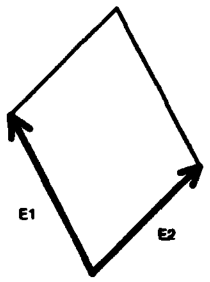


Figure 5: Two Vectors Contained on a Surface

Then:

$$-\Delta z_1 = E_1 \cdot G$$
$$-\Delta z_2 = E_2 \cdot G$$

These can be combined into a single matrix equation, in which the upper row is the first equation above, and the lower row is the second equation:

$$\begin{bmatrix} -\Delta z_1 \\ -\Delta z_2 \end{bmatrix} = \begin{bmatrix} E_1 \cdot G \\ E_2 \cdot G \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} G^T$$

$$G^T = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} -\Delta z_1 \\ -\Delta z_2 \end{bmatrix}$$

This expresses the surface gradient G as a function of the x, y, and z components of two vectors contained on the surface.

As a corollary:

1. Under orthography, vectors E_1 and E_2 have the same coordinates in the image that they have in the scene. So, given the Δz values for two vectors on a surface, the gradient of the surface can be found using the image.

7. Gradients of Perpendicular Vectors and Planes
Suppose two vectors $(\Delta x_1, \Delta y_1, \Delta z_1)$ and $(\Delta x_2, \Delta y_2, \Delta z_2)$ are perpendicular (in the *scene*), and that their gradients (as defined above) are $G_1 = (p_1, q_1)$ and $G_2 = (p_2, q_2)$ (figure 6). Then:

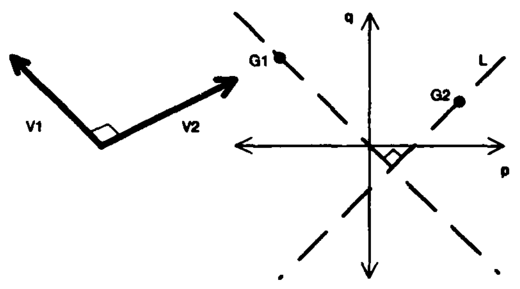


Figure 6: Gradients of Perpendicular Vectors

$$(\Delta x_1, \Delta y_1, \Delta z_1) \cdot (\Delta x_2, \Delta y_2, \Delta z_2) = 0$$
$$\Delta x_1 \Delta x_2 + \Delta y_1 \Delta y_2 + \Delta z_1 \Delta z_2 = 0$$

Dividing by $\Delta z_1 \Delta z_2$,

$$p_1 p_2 + q_1 q_2 + 1 = 0$$
$$G_1 \cdot G_2 = -1$$

Suppose that G_1 is given. Then, the above equation is a line L in gradient space, which is the loci of the possible locations of G_2. In fact,

- L is perpendicular to the line from G_1 to the origin
- the distance from L to the origin is the reciprocal of the distance from G_1 to the origin
- L is on the opposite side of the origin from G_1.

The line L is Huffman's *dual line* (for a line in the image whose gradient is G_1) [5].

This result has two corollaries:

1. If two planes are orthogonal, their gradients obey the above relationship, since their surface normals are perpendicular.
2. If a plane contains a vector, the gradients of the plane and vector obey the above relationship, since the vector is perpendicular to the surface normal.

8. Rotation of the Image and Gradient Space
Suppose a surface is defined by $-z = f(x, y)$ and has gradient (p, q).

If we rotate the x-y axes around the z-axis through some angle θ to new (x'-y') axes (figure 7), then:

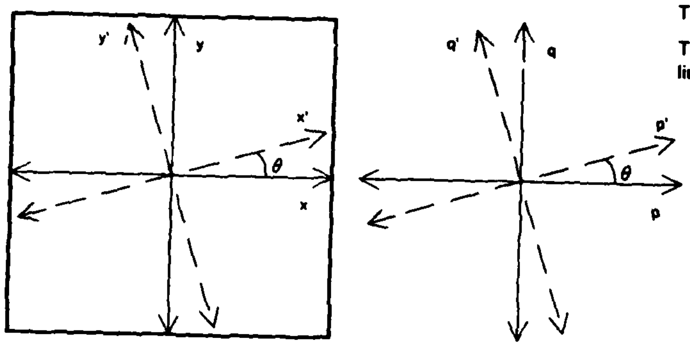


Figure 7: Rotating the Image and the Gradient Space

$$(x, y) = (x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)$$

Now,

$$\begin{aligned} -z' &= -z = f(x,y) \\ &= f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta) \end{aligned}$$

Using

$$\frac{df(x,y)}{du} = \frac{\partial f}{\partial x}\frac{dx}{du} + \frac{\partial f}{\partial y}\frac{dy}{du}$$

we have:

$$p' = \frac{\partial f}{\partial x'} = p\cos\theta + q\sin\theta$$

and

$$q' = \frac{\partial f}{\partial y'} = -p\sin\theta + q\cos\theta$$

This amounts to rotating the p-q axes by θ to determine the p'-q' axes. Thus, rotation of the image corresponds to identical rotation of the gradient space.

Perspective

In this section, *perspective projection* is assumed and its consequences in gradient space are described. While most of the results were presented by Kender [11], some new results are included in sections 12 and 13. This paper does not deal with scene transformations (such as rotation) and their effects in the image under perspective projection [14], nor with camera models [1, 2]. Instead, it describes the relationship between the scene, the image, and gradient space under perspective projection.

9. Vanishing Point of a Line

Suppose a line in the scene is defined by $(x, y, z) + a(\Delta x, \Delta y, \Delta z)$ for all values of a, where (x, y, z) is any point on the line and $(\Delta x, \Delta y, \Delta z)$ is a direction vector of the line (i.e. any vector contained in the line). For any a, the corresponding point on the line is $(x + a\Delta x, y + a\Delta y, z + a\Delta z)$ and its image point P_a is:

$$P_a = \left(\frac{-x - a\Delta x}{z + a\Delta z}, \frac{-y - a\Delta y}{z + a\Delta z}\right)$$

As a grows larger, the image point P_a converges to some point V in the image (figure 8):

$$V = \lim_{a\to\infty} P_a = \left(\frac{-\Delta x}{\Delta z}, \frac{-\Delta y}{\Delta z}\right)$$

The image point V is called the *vanishing point* of the line.

The assumption has been made here that $\Delta z \neq 0$, i.e. that the line is not parallel to the image plane.

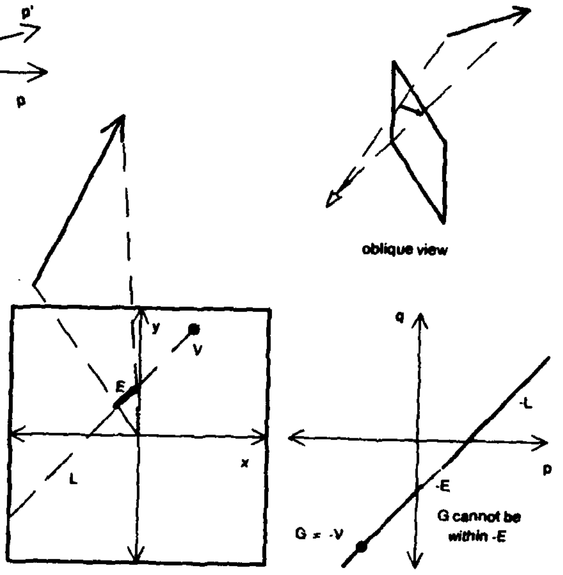


Figure 8: Vanishing Point of a Line

Corollaries of this definition are:

1. If a line (or vector) has gradient G and vanishing point V, then $G = -V$.

2. Since V depends only on the direction vector $(\Delta x, \Delta y, \Delta z)$, parallel lines have the same vanishing point. Thus, each point in the image is the vanishing point for a family of parallel lines.

3. If a line passes through the origin, then its vanishing point is the point at which it intersects the image plane ($z = -1$).

4. The vanishing point V of a vector must lie on the image line L containing the image of the vector (figure 8). The vector's gradient G must therefore lie on the line $-L$ in gradient space, where $-L$ is the line:

   - parallel to L
   - at an equal distance from the origin as L
   - on the opposite side of the origin from L.

Two additional observations concerning G are:

a. The vanishing point of a vector cannot be in the middle of its image, so if E is the image of the vector, then V cannot be within E and G cannot be within $-E$ (figure 8).

b. If the vector in the scene is parallel to the image plane, it has no vanishing point (i.e. V is infinitely far away on L); it has no gradient (i.e. G is infinitely far from the origin, in the direction parallel to L).

10. Vanishing Line of a Surface

Suppose a surface S has gradient G_S. For any vector L on S with gradient G_L, $G_S \cdot G_L = -1$, as shown in section 7. Since the vanishing point of L is $V_L = -G_L$ (by corollary 1 of section 9),

$$G_S \cdot V_L = 1$$

Suppose that G_S is given. Then the above equation defines a line V_S in the image, containing the vanishing points V_L for all vectors L contained on S (figure 9). In fact,

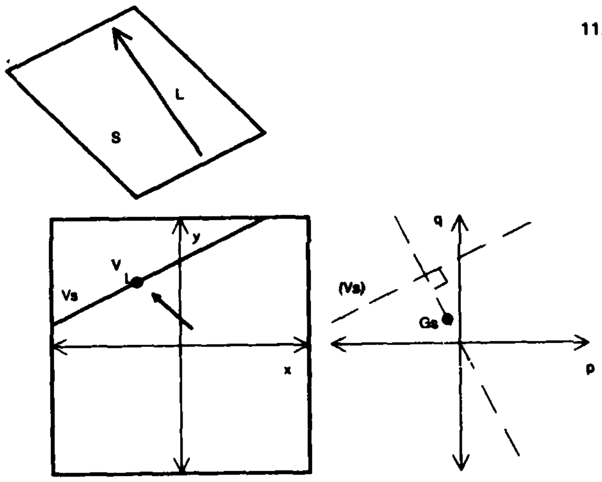


Figure 9: Vanishing Line and Gradient of a Surface

- V_S is perpendicular to the line from G_S to the origin
- the distance from V_S to the origin is the reciprocal of the distance from G_S to the origin
- V_S is on the same side of the origin as G_S.

The line V_S is called the *vanishing line* of the surface; it is the locus of vanishing points for all vectors on the surface.

Corollaries of this definition are:

1. Since V_S depends only on G_S, parallel surfaces have the same vanishing line. Thus, each line in the image is the vanishing line for a family of parallel surfaces.

2. If a surface passes through the origin, its vanishing line is the line along which it intersects the image plane ($z = -1$).

3. Suppose L is a line in the image. There exists a family of parallel surfaces for which L is the vanishing line. These surfaces all have the same gradient, which might be called the *vanishing gradient* for L, denoted G^v_L. Let L be the set of points (x,y) defined by the equation: $1 = ax + by = (a,b) \cdot (x,y)$. Then by section 10, since (x,y) is a spoint on L, (a,b) must be the gradient of the surfaces for which L is the vanishing line, i.e. $G^v_L = (a, b)$. Thus, for any line L in the image, we can determine the associated *vanishing gradient* G^v_L -- the gradient of the surfaces for which L is the vanishing line.

4. Suppose edge E is the image of some vector V with gradient G_V. If S is the surface through the origin and E, then E is the vanishing line of S (by corollary 2 above) and the gradient of S is G^v_E (by corollary 3 above). V must be contained on the surface S, so by corollary 2 of section 7,

$$G_V \cdot G^v_E = -1$$

This is the relationship between a vector and the vanishing gradient of its image.

11. Point Contained on a Surface

Suppose a surface S has gradient $G = (p, q)$ and intersects the z-axis at $z = D$ (i.e. the plane is defined by $px + qy + z - D = 0$). Let $P = (x, y)$ be a point in the image; it must correspond to some point X on surface S in the scene (figure 10).

Since the image of X is P, $X = (ax, ay, -a)$ for some value of a. Since X also lies on S,

$$p(ax) + q(ay) + (-a) - D = 0$$

Solving this equation for a yields $a = D / (px + qy - 1) = D / (P \cdot G - 1)$ and

$$X = \frac{D}{P \cdot G - 1}(x, y, -1)$$

X is sometimes called the *back-projection* of image point P onto surface S [11].

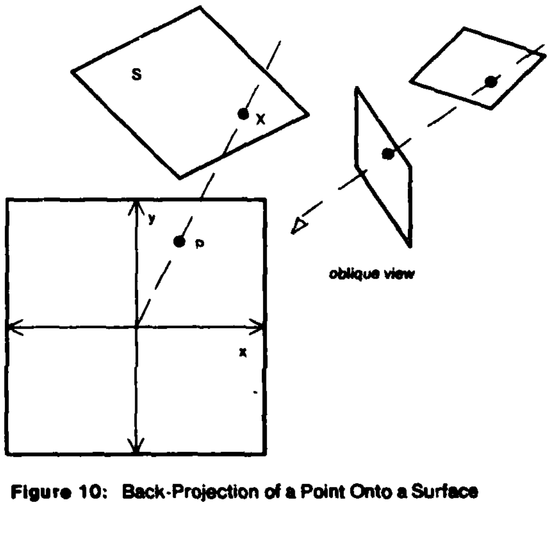


Figure 10: Back-Projection of a Point Onto a Surface

The quantity $D / (P \cdot G - 1)$ is the distance from X to the x-y plane ($z = 0$). If this value is negative, then X is not in the scene (i.e. it is behind the viewer), and P does not correspond to any point on the image of S.

The assumption has been made here that $P \cdot G - 1 \neq 0$, i.e. that P does not lie on the vanishing line of S.

12. Vector Contained on a Surface

Suppose a vector V has gradient G_V and lies on a surface S with gradient G_S. Then by corollary 2 of section 7:

$$G_S \cdot G_V = -1$$

Now suppose that V is visible in the image as some edge E (figure 11). By corollary 4 of section 10, if G^V_E is the vanishing gradient of E,

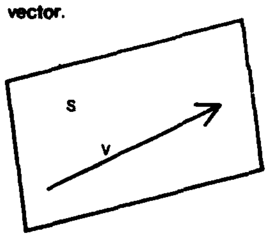
$$G^V_E \cdot G_V = -1$$

These equations can be combined into a single matrix equation, in which the upper row is the first equation, and the lower row is the second equation:

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} G_S \cdot G_V \\ G^V_E \cdot G_V \end{bmatrix} = \begin{bmatrix} G_S \\ G^V_E \end{bmatrix} G_V^T$$

$$G_V^T = \begin{bmatrix} G_S \\ G^V_E \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Since G^V_E is determined by the edge E in the image, this equation relates the gradient G_V of a vector with the image E of the vector and the gradient G_S of a surface containing the vector.



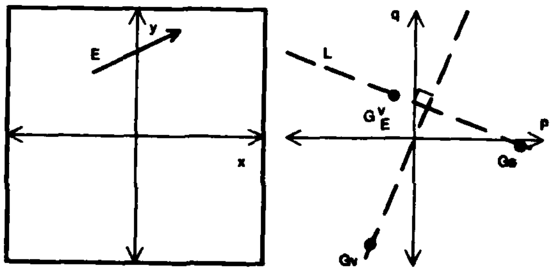


Figure 11: Vector Contained on a Surface Under Perspective

Corollary to this result is:

1. Combining the two equations in a different manner:

$$0 = (G_S \cdot G_V) - (G^V_E \cdot G_V)$$
$$0 = (G_S - G^V_E) \cdot G_V$$
$$(G_S - G^V_E) \perp G_V$$

So, the line L in gradient space containing G_S and G^V_E is perpendicular to the line through the origin and G_V (figure 11).

2. There is an interesting restriction on line L in corollary 1. As shown in corollary 1, L must pass through G^V_E. Its slope depends on the gradient G_V of V. G_V is described in corollary 4 of section 9: it must lie on the *line* $-E$, but not within the *line segment* $-E$ corresponding to the edge itself. This constrains the orientation of L such that the line through G^V_E perpendicular to L cannot pass through the line segment $-E$ (figure 12). Hence, the position and length of an edge in the image constrain the gradients of surfaces containing the corresponding vector in the scene.

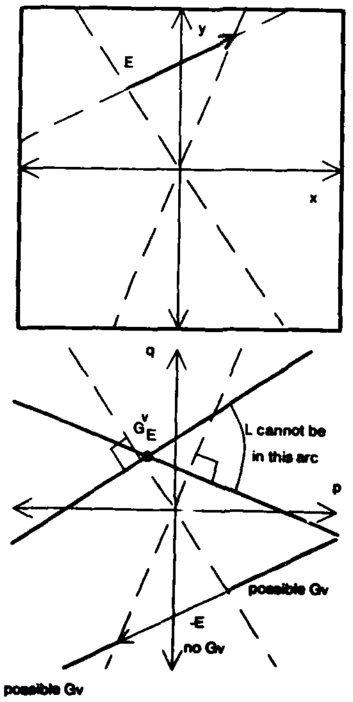


Figure 12: Restrictions on the Slope of L

13. Vector Contained on Two Surfaces

Suppose a vector with gradient G_V is the boundary between two surfaces with gradients G_1 and G_2 (figure 13). If the vector appears as an edge E in the image, and G^V_E is the vanishing gradient of E, then by corollary 1 or result 12,

$$(G_1 - G^V_E) \perp G_V$$
$$(G_2 - G^V_E) \perp G_V$$

So, $(G_1 - G^V_E) \parallel (G_2 - G^V_E)$, i.e. G^V_E, G_1, and G_2 are collinear in gradient space.

Since G^V_E was determined by the location of the edge E in the image, the constraint provided on G_1 and G_2 is that they lie on a line which passes through G^V_E. This line is the same as L in corollaries 1 and 2 of section 12, and its oritneation is limited to certain angles depending on the location and size of the image edge E.

This is the *connect edge* relation under perspective; it is the perspective counterpart to corollary 1 of section 5.

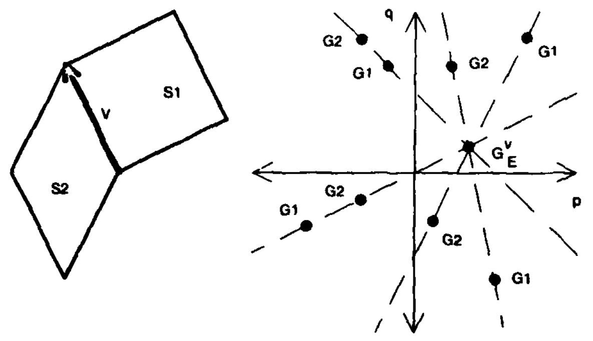


Figure 13: Vector Contained on Two Surfaces Under Perspective

Curved Surfaces and Arcs

In these sections, the gradient is defined for an arc in the scene. Then, using calculus, the fundamental results are developed concerning curved arcs and surfaces. The results are very similar to those of sections 4, 5, and 6; this is because surface and arc gradients capture the same (first-order differential) information contained in tangent lines and planes, which obey the results of sections 4, 5, and 6.

14. Gradient of an Arc

Suppose an arc A is defined (in parametric form) by $(x(s), y(s), z(s))$. Then its gradient can be defined as:

$$G_A = (p,q) = (\frac{dx}{dz}, \frac{dy}{dz}) = (\frac{dx}{ds}\frac{ds}{dz}, \frac{dy}{ds}\frac{ds}{dz})$$

Note that both p and q are themselves functions of s.

Corollaries:

1. At any point on an arc defined as above, the tangent vectors are defined by:

$$T = a(\frac{dx}{ds}, \frac{dy}{ds}, \frac{dz}{ds}) \quad \text{for all } a$$

2. Thus, the gradient G_A of an arc A is the same as the gradient G_T of a tangent vector T to the arc (figure 14).

3. If the gradient of an arc is (p, q), then the vector $(p, q, 1)$ is tangent to the arc.

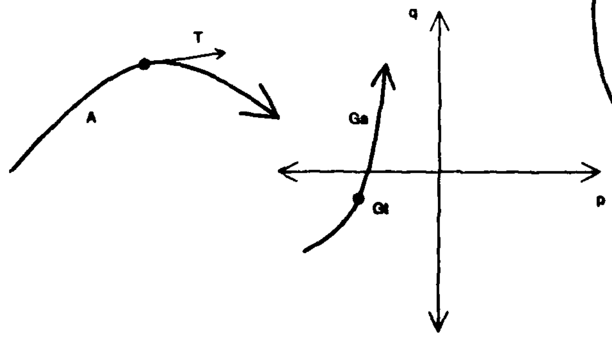


Figure 14: Gradient of an Arc and a Tangent Vector

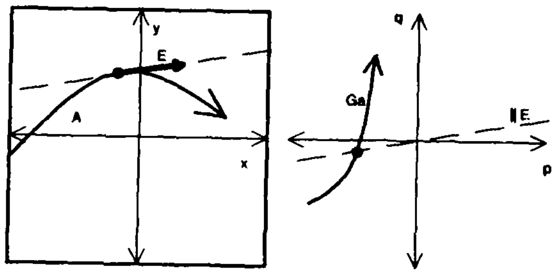


Figure 15: Gradient of an Arc Under Orthography

4. Under orthography, the tangent vector to an arc is visible as the edge $E = a\,(dx/ds, dy/ds)$ for some value of a (figure 15). If the gradient of the arc is $G = (dx/dz, dy/dz)$, then $G = (E/a)\,(ds/dz)$. In other words, the line in gradient space from the origin to G is parallel to the tangent vector E seen in the image.

15. Arc Contained on a Surface

Suppose an arc $A = (x_A(s), y_A(s), z_A(s))$ is contained on a surface S defined by $-z = f(x, y)$. Let the arc gradient be $G_A = (p_A, q_A) = (dx_A/dz_A, dy_A/dz_A)$, and let the surface gradient be $G_S = (p_S, q_S) = (\partial f/\partial x, \partial f/\partial y)$. Then for all s,

$$-z_A(s) = f(x_A(s), y_A(s))$$

We can differentiate using the rule:

$$\frac{df(x,y)}{ds} = \frac{\partial f}{\partial x}\frac{dx}{ds} + \frac{\partial f}{\partial y}\frac{dy}{ds}$$

to yield:

$$-\frac{dz_A}{ds} = \frac{\partial f}{\partial x}\frac{dx_A}{ds} + \frac{\partial f}{\partial y}\frac{dy_A}{ds}$$
$$= G_S \cdot (\frac{dx_A}{ds}, \frac{dy_A}{ds})$$

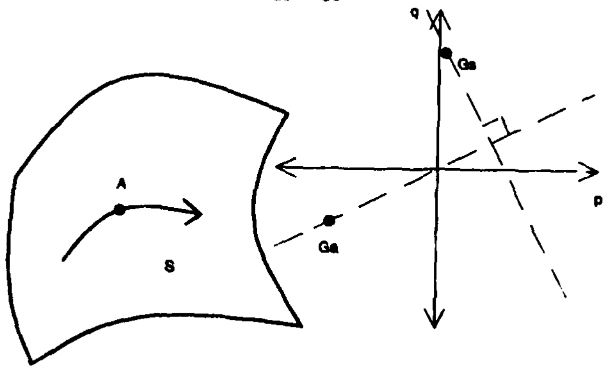


Figure 16: Arc Contained on a Curved Surface

Corollaries to this result are:

1. With the above definitions, the above equation can be multiplied by ds/dz_A to yield:

$$-1 = G_S \cdot G_A$$

So, G_A and G_S obey the relationship of section 7 (figure 16).

2. If $(\Delta x, \Delta y, \Delta z)$ is tangent to arc A at point X, and A is contained on a surface whose gradient at X is G, then $-\Delta z = G \cdot (\Delta x, \Delta y)$. Thus, the results of section 4 apply to the tangent vector to an arc.

16. Arc Contained on Two Curved Surfaces

Suppose an arc $A = (x(s), y(s), z(s))$ with gradient G_A is the boundary between two surfaces S_1 and S_2 with gradients G_1 and G_2 (figure 17). At any point on A, we have (by corollary 1 of section 15):

$$-1 = G_A \cdot G_1 = G_A \cdot G_2$$
$$0 = G_A \cdot (G_1 - G_2)$$
$$G_A \perp (G_1 - G_2)$$

So the line containing G_1 and G_2 in gradient space is perpendicular to the line from the origin to G_A.

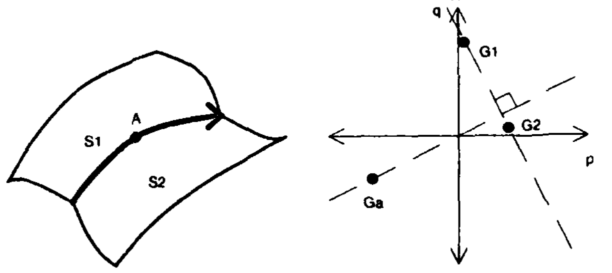


Figure 17: Arc Contained on Two Curved Surfaces

Corollary to this is:

1. Under orthography, we can combine the above result with corollary 3 of section 14 to conclude that the line containing G_1 and G_2 in gradient space is perpendicular to the tangent to the arc in the image. This is the counterpart to the *connect edge* relationship for curved surfaces and arcs.

17. Two Arcs Contained on a Curved Surface

Suppose arcs A_1 and A_2 with gradients G_1 and G_2 are contained on a surface with gradient G_S (figure 18).

At a point of intersection of A_1 and A_2, we have (by corollary 1 to section 15):

$$-1 = G_1 \cdot G_S$$
$$-1 = G_2 \cdot G_S$$

These can be combined into a single matrix equation to yield:

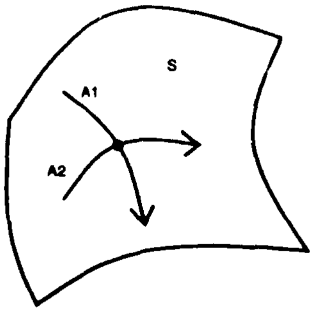


Figure 18: Two Arcs Contained on a Curved Surface

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} G_S^T$$

$$G_S^T = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

This allows us to compute the gradient of a surface from the gradients of two intersecting arcs on the surface.

Corollary to this is:

1. Under orthography, suppose that edges E_1 and E_2 in the image are tangent to the images of arcs A_1 and A_2 at a point where they intersect, and that E_1 and E_2 correspond to scene vectors $(\Delta x_1, \Delta y_1, \Delta z_1)$ and $(\Delta x_2, \Delta y_2, \Delta z_2)$. If A_1 and A_2 are contained on a surface with gradient G_S, then:

$$-\Delta z_1 = E_1 \cdot G_S$$
$$-\Delta z_2 = E_2 \cdot G_S$$
$$G_S^T = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} -\Delta z_1 \\ -\Delta z_2 \end{bmatrix}$$

Thus, under orthography, the gradient of a surface can be computed from the Δz values for two vectors tangent to arcs on the surface, at a point where the arcs intersect.

Summary

In this paper, we have defined the gradient for surfaces, planes, arcs, and vectors, and we have seen that the gradient and Δz attribute of an object are mutually constrained. We have also seen that knowledge about the gradient (or Δz-component) of a surface can be used to determine the gradient of a vector or arc on that surface, and that knowledge about the gradients of two such vectors or arcs can be used to uniquely determine the surface gradient. In addition, features in the image can be combined with gradient or Δz information to yield three-dimensional reconstructions of scene objects, under both perspective and orthographic projections.

This collection of theorems and definitions includes a recounting of results from Huffman [5], Mackworth [12], and Kender [11] as well as some new notation and results. The definition and use of *arc* and *vector* gradients as well as *surface* gradients has provided a more concise notation for several of these results.

Bibliography

[1] Forseth, K.
Graphics for Architecture.
Van Nostrand Reinhold, New York, 1980.

[2] Haralick, R. M.
Using Perspective Transformations in Scene Analysis.
Computer Graphics and Image Processing 13:191-221, 1980.

[3] Horn, B. K. P.
Understanding Image Intensities.
Artificial Intelligence 8:201-231, 1977.

[4] Horn, B. K. P. and Schunck, B. G.
Determining Optical Flow.
Artificial Intelligence 17:185-203, 1981.

[5] Huffman, D. A.
Impossible Objects as Nonsense Sentences.
In Meltzer, B. and Michie, D. (editor), *Machine Intelligence 6*, chapter 19pages 295-323. American Elsevier Pub. Co., New York, 1971.

[6] Huffman, D. A.
Realizable Configurations of Lines in Pictures of Polyhedra.
In Meltzer, B. and Michie, D. (editor), *Machine Intelligence 8*, chapter 24pages 493-509. American Elsevier Pub. Co., New York, 1977.

[7] Kanade, T.
A Theory of Origami World.
Artificial Intelligence 13:279-311, 1980.

[8] Kanade, T. and Kender, J.
Mapping Image Properties into Shape Constraints: Skewed Symmetry, Affine-Transformable Patterns, and the Shape-from-Texture Paradigm.
In *IEEE Workshop on Picture Data Description and Management*, pages 130-135. Asilomar, CA, August, 1980.

[9] Kanade, T.
Recovery of the Three-Dimensional Shape of an Object from a Single View.
Artificial Intelligence 17:409-460, 1981.

[10] Kender, J.
Shape From Texture: A Computational Paradigm.
In Baumann, L. S. (editor), *Proc. ARPA IUS Workshop*, pages 134-138. April, 1979.

[11] Kender, J. R.
Shape from Texture.
PhD thesis, Carnegie-Mellon University, Computer Science Department, November, 1980.

[12] Mackworth, A. K.
Interpreting Pictures of Polyhedral Scenes.
Artificial Intelligence 4:121-137, 1973.

[13] Mackworth, A. K.
Use of Constraints for Interpreting Three-Dimensional Scenes.
PhD thesis, University of Sussex, 1974.

[14] Newman, W. M. and Sproull, R. F.
Principles of Interactive Computer Graphics, 2nd ed.
McGraw-Hill, New York, 1979.

[15] Shafer, S. A. and Kanade, T.
Using Shadows in Finding Surface Orientations.
Technical Report, Carnegie-Mellon University, Computer Science Department, January, 1982.
Submitted to Computer Graphics and Image Processing.

[16] Woodham, R. J.
A cooperative algorithm for determining surface orientation from a single view.
In *Proc. 5th IJCAI*. 1977.

AD-P000 134

LINEAR DELINEATION

Martin A. Fischler

SRI International, Menlo Park, California 94025

ABSTRACT

In this paper we address the most general version of the problem of achieving machine perception and tracing of "line-like" structures appearing in an image; our goal is human level performance. We show that the problem can profitably be viewed as the process of finding skeletons in a gray scale image after observing (1) that line detection does not necessarily depend on gradient information, but rather is approachable from the standpoint of measuring total intensity variation, and (2) that smoothing the original image produces an approximate distance transform. We present an effective technique for extracting the delineating skeletons from an image, and show examples of this approach employing aerial, industrial, and radiographic imagery.

I INTRODUCTION

For many tasks in scene analysis, there may not exist general solutions independent of purpose or intended application. However, for the task of linear delineation, one can easily find image subsets for which a panel of human observers would be almost unanimous in their interpretation without having to agree on the explicit criteria underlying their decisions; our goal is to produce a computer system that can perform the delineation task at close to human levels for at least these more obvious cases (especially where semantic knowledge is not required). In this paper we present some new ways of looking at the problem of linear delineation and provide techniques that are significantly more general and effective than previously reported methods for this task.

II PROBLEM DEFINITION

For the purposes of this paper, we define linear delineation (LD) as the task of generating a set of lists (of coordinates) of points, for a given 2-D image, such that the points in each list fall sequentially along what any reasonable human observer would describe as a clearly visible "line-like" structure in the image. Practical examples of this task might be to delineate the roads, rivers, and rail lines in an aerial photograph, or to trace the paths taken by blood vessels in a radiographic angiogram, or to locate the wiring paths on a printed circuit board; however, our goal in this paper is not to look for specific real-world objects or to assign semantic labels to the detected linear structures, but rather to find the most perceptually obvious (to a human observer) occurrences of such structures. We further distinguish between the problems of (1) detecting the edges or contours of extended objects, and (2) delineating those objects whose appearance is adequately represented by a central skeleton -- we only address the second problem.

III LINES AND EDGES

While most approaches to LD do not distinguish between lines and edges, and even use edge detection as a necessary first step in the delineation task, a critical concept advanced in this paper is the distinction between line and edge detection.

Edge detection is intuitively based on the concept of a discontinuity in intensity (or other locally measurable attribute such as color or texture) between two adjacent but distinct regions in an image. However, in a digital representation of an image, we can always fit a smooth surface to the sample values of the integer raster. Thus, edge detection must be based on parameters or thresholds set by assumptions about the nature of the image. (Even if we only mark as edge points those locations at which there are first derivative maxima, or zeros of the second derivative, we must still ultimately make an arbitrary decision in deciding when the corresponding gradient is large enough to be called a discontinuity.)

Intuitively, a "line-like" or linear structure is a (connected) region that is very long relative to its width, and has a ridge or skeleton along which the intensities change slowly and are distinguished from those outside the region; the width need not be constant, but any changes in width should occur in a smooth manner. (To simplify our discussion, we will assume the linear structures are distinguished by their ridge points being brighter than the surrounding background; but any other specified attribute, which is locally detectable, would be an acceptable substitute.) It is important to recognize the fact that a clearly visible line in an image may have no locally detectable edges (and thus no locally measurable width), or possibly only one detectable edge, or even two edges which are significantly separated and nonparallel (e.g., as might occur in a local widening of a river). Finally, it will generally be the case that linear structures have no visible internal detail that is essential to their delineation.

As a point of interest, it might be noted that the mechanisms for generating subjective edge and line illusions are quite different; subjective edges appear to require a 3-D interpretation, while subjective lines appear to be produced by adaptation phenomena.

IV SMOOTHING, DISTANCE TRANSFORMS, AND THE GRAY SCALE SKELETON

If we can find the edges of a linear structure (LS), we can generate a distance transform and extract a skeleton as the desired delineation (e.g., Rosenfeld [1], Fischler [2]). However, as we noted in the preceding section, LS do not always have locally detectable edges, and, since all of the generally known techniques for deriving a skeleton require a complete contour, some other approach is required. (The classical skeletonizing techniques intimately link the contour/edges of a region and its skeleton, and it is just this linkage that we wish to break).

Surprisingly, something equivalent to a distance transform that works on gray scale images (and on binary images as well) is already available. To achieve our purpose, we need only observe that the intensities in a properly smoothed image can be considered to be the values of an approximate distance transform. What is the best smoothing function for general use? Actually, it doesn't seem to make much difference in many cases. Most digital images have been processed by low-pass optical and electronic systems that have inserted the required minimum level of smoothing. The viewpoint that the smoothed image can be considered to be a distance transform is the essential element. If we start with a binary image, or a very noisy image, then additional smoothing is desirable and possibly even necessary. Since we are not concerned with blurring edges, and we would like to eliminate (blur) any structure or texture internal to the linear regions, we want the smoothing function to have a width of at least 1/2 that of the region to be delineated. If we increase the width of the smoothing function, we eventually eliminate the thinner linear structures, and thus, if we wish to find all possible LS without prior knowledge of the content of the image, the processing should be repeated with a set of filters having a spectrum of widths. Actually, no more than 1 or 2 filtering steps should ever be required. For example, to trace all the linear structures (diameters up to 20 pixels) in a noisy radiographic angiogram, a single filter of width 10 was used (see Figure 1). In a 256X256 image, we could probably trace all the linear structures with a diameter of up to approximately 1/4 of the image width by employing just two filters (widths 10 and 30). To trace the roads in an aerial image, the filtering produced by reducing the image from 1024X1024 to 256X256 was sufficient to achieve excellent results (see Figure 4).

V RIDGES (OR VALLEYS), OPERATORS, AND NEIGHBORHOODS

Having produced an approximate distance transform via smoothing, we now must deal with the problem of locating the ridge points that denote the spines (skeletons) of the LS. When an exact distance transform is derived from a complete contour, noise is not a problem and the skeleton has assured geometric properties that make it easy to detect; finding the ridge points of an approximate distance transform is considerably more complex.

We traditionally distinguish between locally and globally detectable features: local features are detectable by an intensity pattern which can be observed through a small peep-hole centered on the feature, while global features are ambiguous in a small area. The model or description of the local feature is generally compiled into an intensity patch (matched filter or operator) which can be convolved with the image to detect the corresponding feature. In the case of an exact distance transform, a 3X3 pixel operator is sufficient to detect ridge points (a 2X2 operator is sufficient for the Labeled Distance Transform (Fischler [2])); for the approximate distance transform, a small fixed-size operator is ineffective.

The principal utility of a local operator is that the number of data patterns the operator might encounter is small enough to allow one to enumerate a decision for each such pattern. If we further agree to use a small square window of the image as our local domain, and to use either table look-up or convolution as the basis for decision making, then a uniform mechanization can be employed to implement a large number of distinct (generally unrelated) local operators. The attractiveness of this second (implementation) aspect has led to the situation that almost all low-level (local) scene analysis is done using such peep-hole type operators. The disadvantage of this approach is that the concept of local is relative to the size of the entity of interest, and either one must know this size in advance, or use a whole family of operators of increasing size, where the larger operators loose the advantages that led to their use in the first place. In the case of line detection, where the line width can vary over a wide range of values, the conventional operator concept is inappropriate.

Based on these general issues (even more so than on the immediate problem at hand), we have considered other realizations of general "local" decision-making processes that satisfy the previously stated conditions, but do not necessarily lend themselves to a convolution type mechanization. In particular, restricting our attention to finding the maxima and minima of functions of the displacement along a space curve defined over the image, satisfies our requirements for computational and decision-making simplicity even when the curve traverses the entire image. While the space curve might assume any shape (e.g., follow the contour of an object), the analysis itself is independent of the shape; for the linear delineation problem, we used image intensity as a function of displacement along horizontal and vertical scan lines. Since maxima and minima are symmetrical attributes, we will only discuss the problem of labeling maximal points along the curve.

The problem of finding the ridge points of an approximate distance transform can be viewed as the problem of finding the ridge points (local maxima) of an exact distance transform to which some amount of noise has been added. We are not concerned about the possibility of making isolated (incoherent) incorrect decisions, since, in the next section, we will describe powerful linking and pruning methods capable of eliminating such errors and even eliminating the valid as well as invalid weak coherent structures that are detected. Our main problem is that we can't count on finding either large local gradients or using known line width to determine some minimum significant gradient threshold, to identify valid ridge points, additionally, noise will introduce many false local maxima. Thus, we must use total intensity change, rather than rate of change, to detect valid ridge points; and we must have an effective way of determining such total change even in the presence of local variation introduced by noise. (While it is not immediately obvious that total intensity change, rather than rate of change, will recover the perceptually obvious linear features, our experiments indicate that this is indeed the case.)

Our approach is to detect two distinct types of intensity maxima along the space curves (in this case, horizontal and vertical scan lines), to which we assign the designations "local" and "global" maxima. The local maxima measure of a point is the total intensity difference from the point to the highest of its immediate left and right minima. The left (right) global maxima measure of a point is the total intensity difference from the point to the lowest value found moving to the left (right) prior to encountering a point with an intensity value equal to, or greater than, that of the given point; the global maxima measure of the point is the smaller of its left and right global measures. If a point has an immediate neighbor with a higher intensity value, it is not a maximal point and it is not assigned either a local or global value (actually, for implementation purposes, it is assigned a zero value); on the other hand, every maximal point will have both a local and global value where the global value equals or exceeds the local value.

We have been proceeding under the assumption that a large local intensity maxima (LIM) denotes a significant event, but, in the presence of large variations in image intensity or noise, the global intensity maxima (GIM) would be a better detector of significant intensity variation; however, in a well-smoothed or relatively noise-free image, there might be very little difference in the information contained in the LIM and GIM measures. There is also the issue of deciding what is a large-enough value, of either the LIM or GIM, to indicate significance. In the example shown in Figure 2, only 1/3 of the points were maximal points, and, in a smoothed image, this percentage should be much smaller. Given the powerful linking and pruning techniques we describe in the next section, it might be possible to return all the maximal points in a binary mask and still extract the desired line structure from the background noise contained in such a mask. However, it makes much more sense to first eliminate those maximal points that do not have enough intensity variation to be perceptually distinguishable from a flat background. (It would even appear that we could, without loosing essential information, eliminate those maximal points with a total variation less than that required to perceive them against a random-noise field with the same statistical variation as the measured variation over some surrounding neighborhood in the image.)

We are still working on the question of what constitutes an optimal threshold for maximal point intensity variation (in order to extract a binary mask to represent the major linear structures in the image) and we expect that the ideas mentioned above will soon lead to an acceptable solution. Our optimism is based on the fact that we have been able to use one fixed pair of threshold values (20 intensity units out of 256 for the LIM, and 40 for the GIM) for all of our experiments and still generally obtain very good results (for example, see Figures 1-5); thus, we already have a heuristic solution that avoids the need for manual intervention, and there appears to be a scientific basis for obtaining a more effective answer.

VI CLUSTERING, LINKING, PRUNING, RANKING, AND FINAL DELINEATION

(This work will be described in detail in a paper currently nearing completion: Fischler and Wolf [3])

Based on the availability of a high quality binary overlay depicting the locations of the major linear structures contained in the given gray scale image, obtained as described in the preceding section, we have been able to demonstrate that:

(1) The linking step in the delineation process can effectively be based on the single attribute of geometric proximity, and that a clustering or association step, followed by the construction of a

Minimum Spanning Tree (MST) through the points of each cluster (Fischler [4]), will correctly link the ridge points along the skeletons of the linear structures (see Figures 4 and 5).

(2) The desired delineations will be embedded in trees containing additional branches that are either minor linear structures or noise, and that simple pruning techniques can eliminate most of this unwanted detail (see Figure 3; note that tree pruning can effectively achieve simplifications that would be difficult, if not impossible, at lower levels of organization of the information).

(3) Having properly linked the ridge points and pruned some of the smaller branches of the resulting trees, we can extract long coherent paths by a decision procedure applied at each node of each tree. This decision procedure, based on the (local to the node) branch attributes of intensity, connectivity, and directionality, assigns path connectivity through a node by splitting off incompatible branches; any remaining ambiguities (more than two branches entering a node) are resolved by choosing those pairings that result in the longest paths. (See Figure 3)

(4) The paths obtained in the tree partitioning step can be rank ordered with respect to perceptual quality by a metric based on the path attributes of total length, average intensity, local intensity variation, and continuity (i.e., ratio of total internal gap length to path length). (See Figure 4)

VII CONCLUDING COMMENTS

We have presented the viewpoint that the problem of delineating the obvious linear structures in an image is distinct from that of finding edges or contours, and is best viewed as the process of finding skeletons in gray scale images (i.e., that line detection does not necessarily depend on gradient information, but rather is approachable from the standpoint of detecting total intensity variation); as a necessary step in this process, we have suggested that an approximate gray scale distance transform can be attained by smoothing the original image. We have described an effective technique for finding ridge points (points on the delineating skeleton), and, in the process, raised some important questions about the conventional approach to designing "local operators."

Working with both the binary overlay (produced as discussed above) and the original gray scale image, we have demonstrated via examples that the remaining steps in the delineation process can be effectively achieved.

Our goal in this work has been to approach human levels of performance in finding perceptually obvious delineations in images selected at random from a reasonably broad class of scene domains, and without any human intervention or prior knowledge about the image content. We believe that this goal can be achieved through extension and refinement of the techniques described in this paper.

VIII ACKNOWLEDGEMENT

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract No. MDA903-79-C-0588, and by the National Science Foundation under Contract No. ECS-7917028. The author gratefully acknowledges the important contributions made by Helen C. Wolf to the work described in this paper.

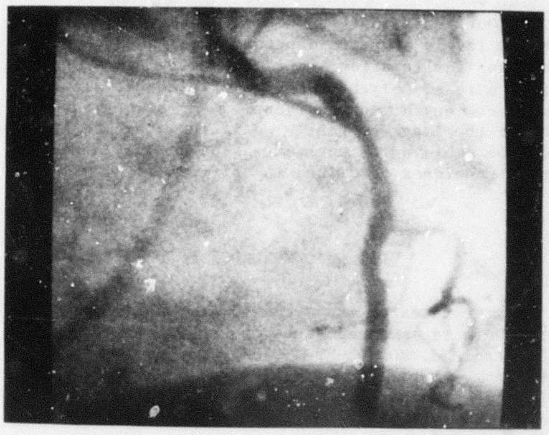
REFERENCES

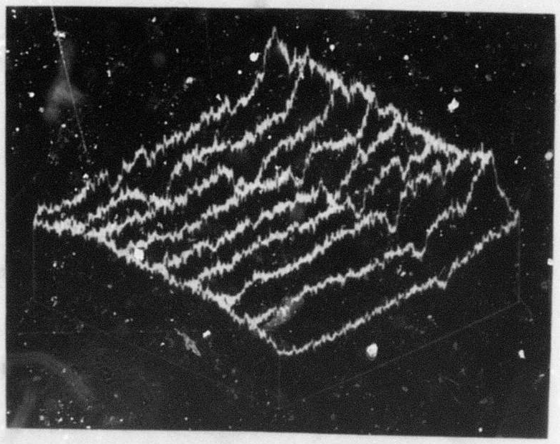
1. Rosenfeld, A. and J. L. Pfalz, "Sequential operations in digital picture processing," J. ACM, vol. 13(4), October 1966, pp. 471-494.

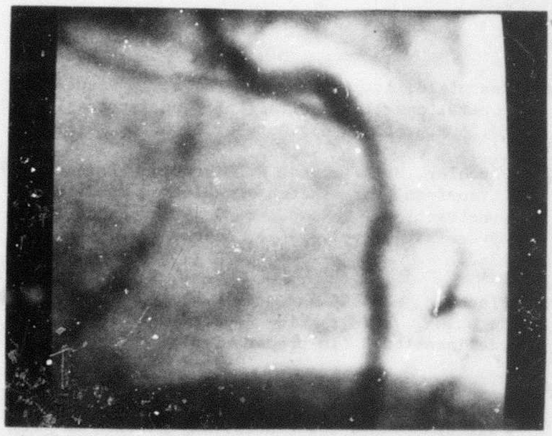
2. Fischler, M. A. and P. Barrett, "An iconic transform for sketch completion and shape abstraction," Computer Graphics and Image Processing, vol. 13(4), August 1980, pp. 334-360.

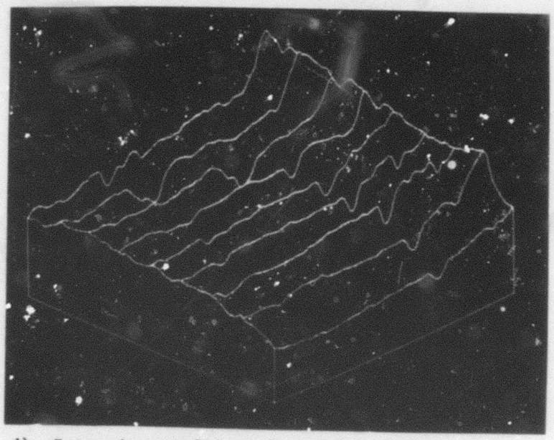
3. Fischler, M. A. and H. C. Wolf, "Linking, pruning, and ranking operations in linear delineation," (in preparation).

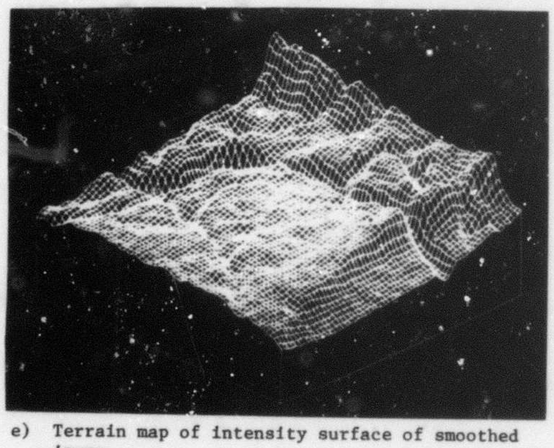
4. Fischler, M. A., J. M. Tenenbaum, and H. C. Wolf, "Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique," Computer Graphics and Image Processing, vol. 15(3), March 1981, pp. 201-223.


a) Original gray scale image


b) Intensity profiles of original image


c) Smoothed gray scale image


d) Intensity profiles of smoothed image


e) Terrain map of intensity surface of smoothed image

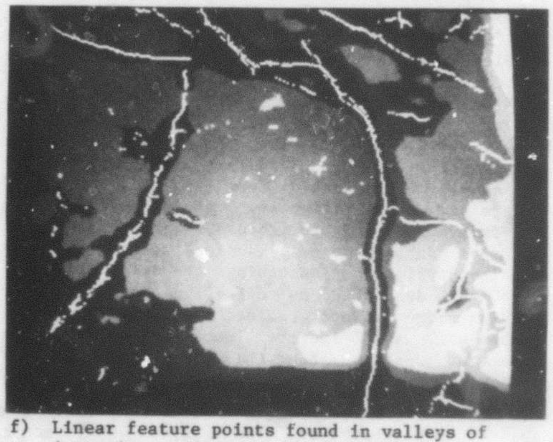
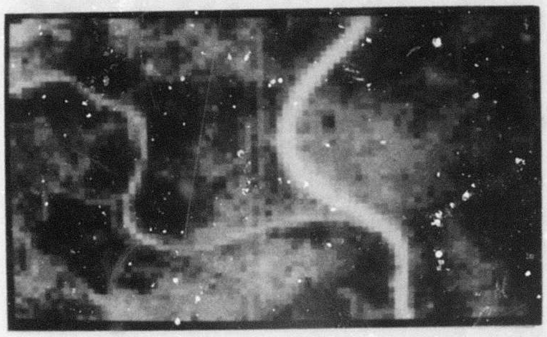
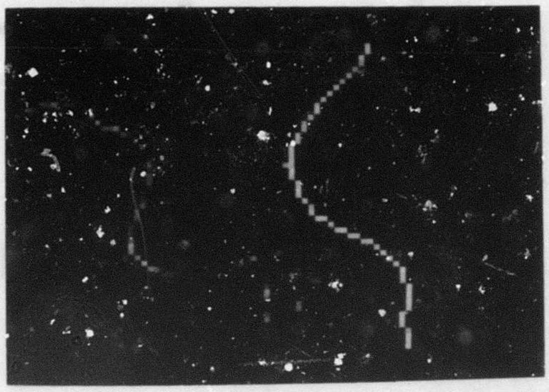

f) Linear feature points found in valleys of intensity surface

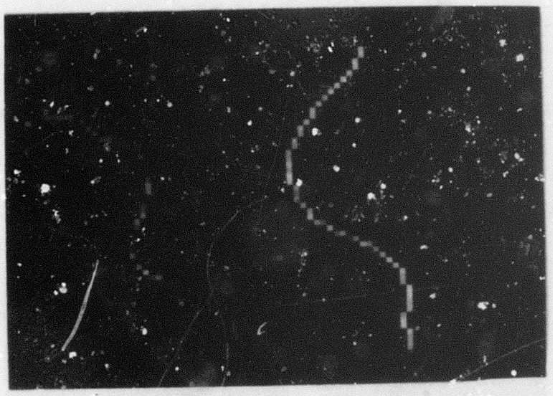
FIGURE 1 -- IMAGE SMOOTHING, AND FINDING LINEAR FEATURE POINTS IN THE INTENSITY VALLEYS OF A RADIOGRAPHIC IMAGE (ANGIOGRAM)



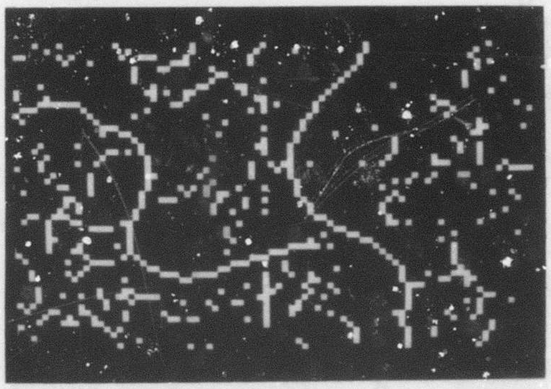
a) Original gray scale image



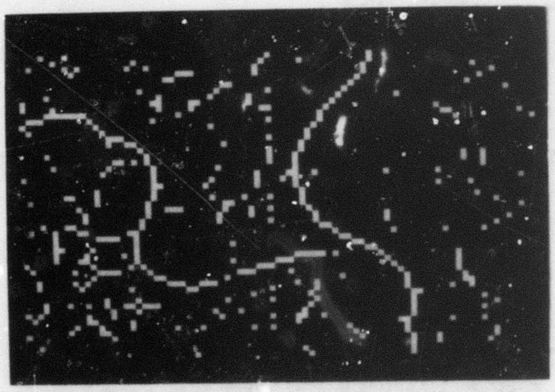
b) Location and magnitude of global intensity maxima



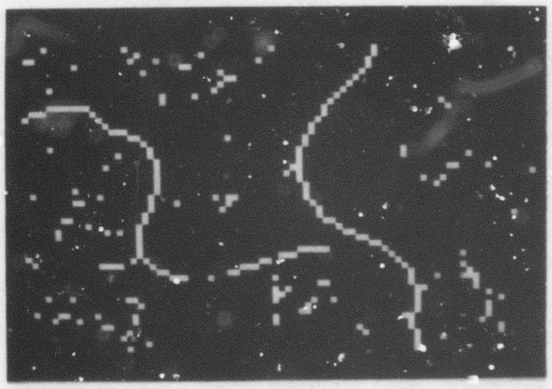
c) Location and magnitude of local intensity maxima



d) Global intensity maxima with magnitude greater than 20

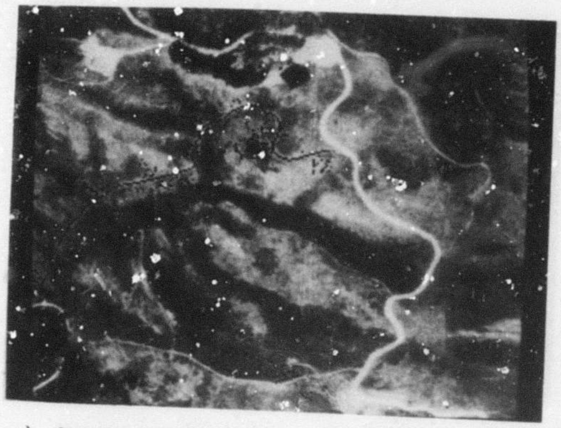


e) Local intensity maxima with magnitude greater than 20

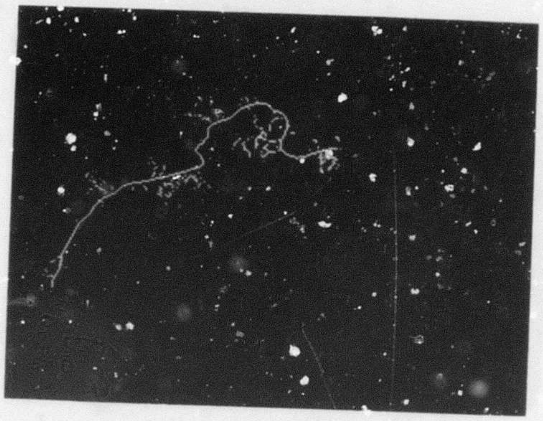


f) Global intensity maxima with magnitude greater than 40

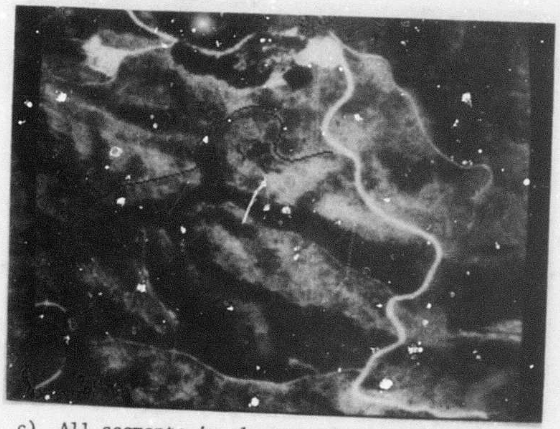
FIGURE 2 -- FINDING LINEAR FEATURE POINTS



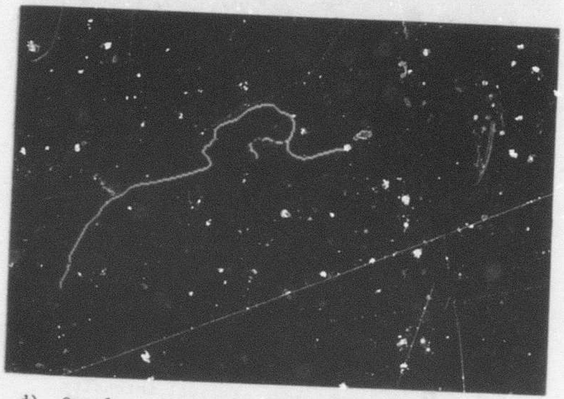
a) One cluster of linear feature points



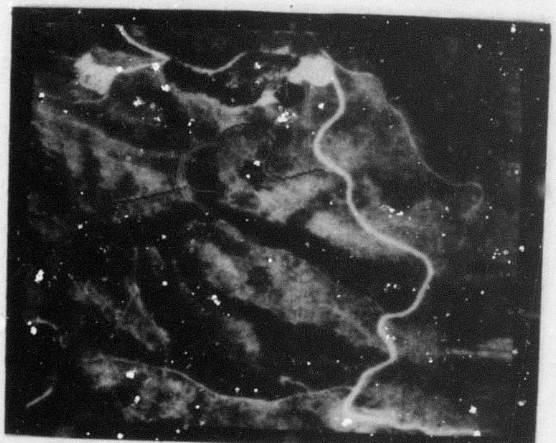
b) Overlay of one cluster of linear feature points



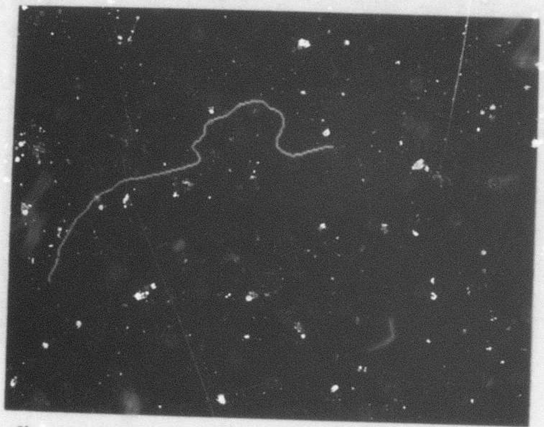
c) All segments in cluster after trimming steps



d) Overlay of all segments in cluster after trimming steps

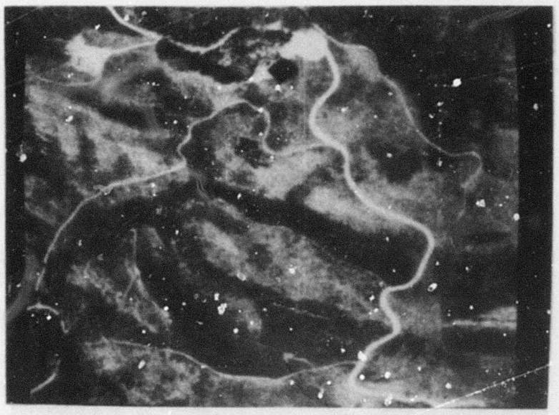


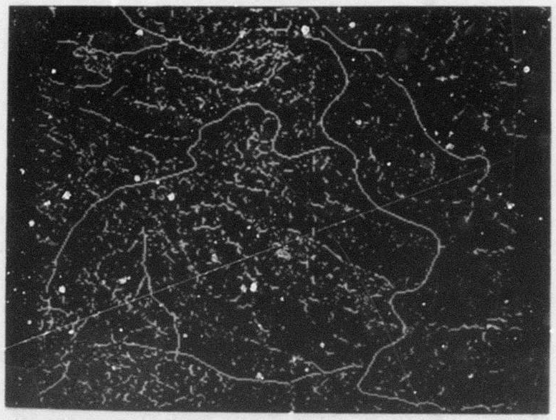
e) Maximum length segment extracted from trimmed cluster

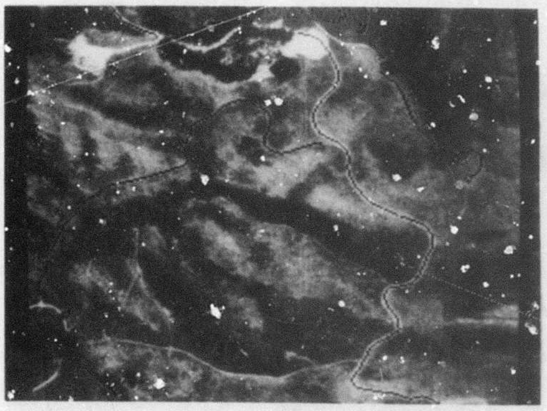


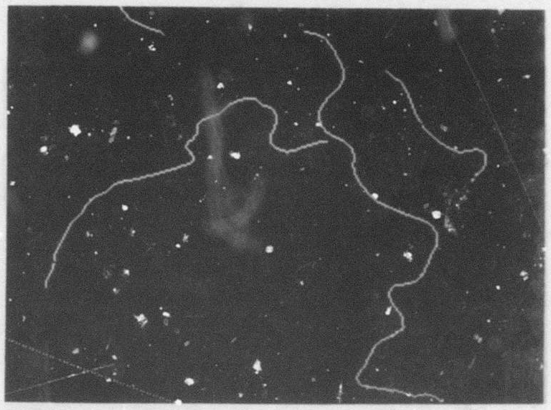
f) Overlay of maximum length segment extracted from trimmed cluster

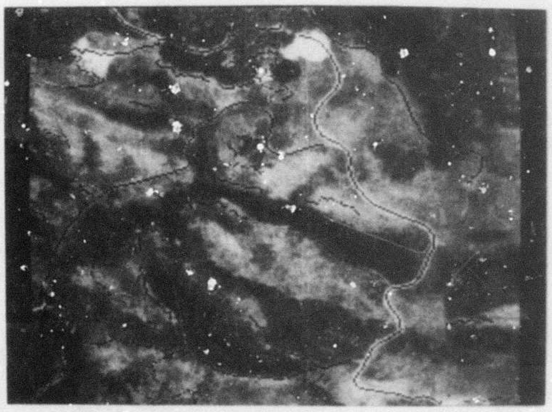
FIGURE 3 -- CLUSTERING, PRUNING, AND LINEAR PATH EXTRACTION


a) Original gray scale picture


b) Overlay of detected linear feature points


c) Four best segments found in linear feature points


d) Overlay of four best segments found in linear feature points


e) All segments found in linear feature points

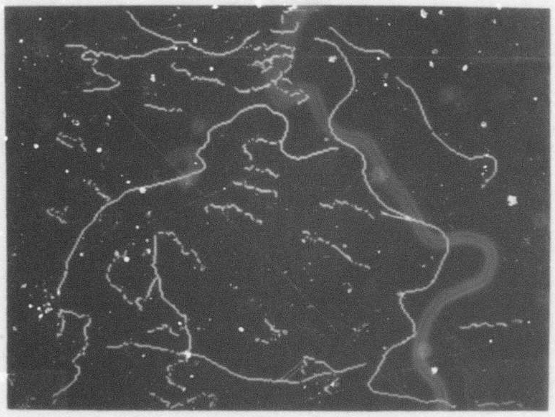
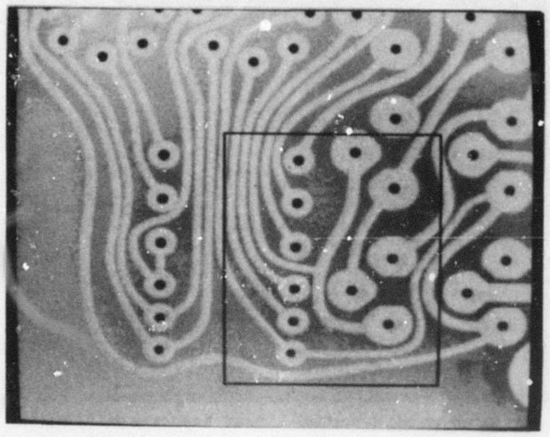
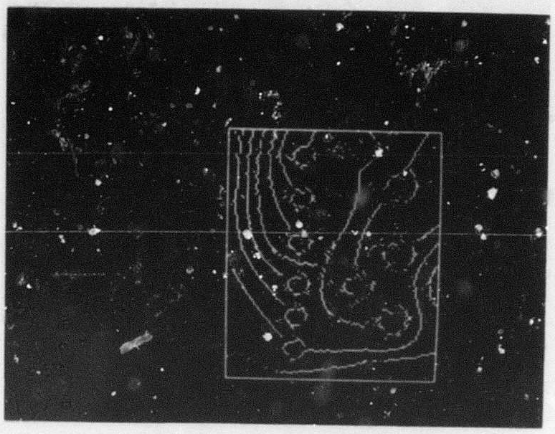

f) Overlay of all segments found in linear feature points

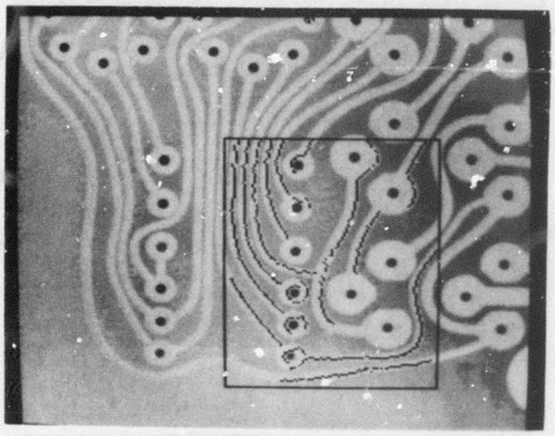
FIGURE 4 -- CLUSTERING, LINKING, AND LINEAR PATH EXTRACTION IN AN AERIAL IMAGE



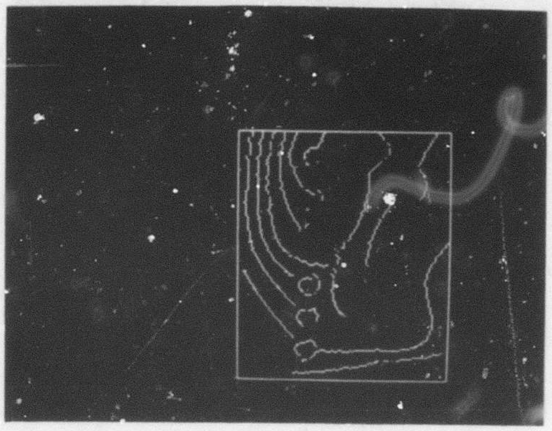
a) Original gray scale picture
(data area enclosed in box)



b) Overlay of detected linear feature points



c) All segments found in linear feature points



d) Overlay of all segments found in linear feature points

FIGURE 5 -- CLUSTERING, LINKING, AND LINEAR PATH EXTRACTION IN AN INDUSTRIAL IMAGE (PRINTED CIRCUIT BOARD)

AD-P000 135

DETECTION OF ERRORS IN MATCH DISPARITIES

Marsha Jo Hannah

Lockheed Palo Alto Research Laboratory
Department 52-53, Building 201
3251 Hanover Street
Palo Alto, CA 94304

ABSTRACT

Most processes that use the disparities resulting from stereo image matching make the unrealistic assumption that every match obtained was correct. Unfortunately, match errors can occur for a variety of image- and terrain-related reasons. Such errors produce undesirable effects when the disparity data is further processed (for instance, to form a scene model), and should be detected and corrected as early in the processing as possible.

Algorithms have been developed to detect errors in disparities derived from aerial photographs of terrain. These algorithms constrain the allowable magnitude of the second derivatives of disparities in local areas around each point. Relaxation-like techniques are employed in the iteration of the detection process.

INTRODUCTION

Most stereo matching processes assume that every match obtained will be a correct match. Unfortunately, this assumption is unrealistic. Many conditions can cause false matches, including low image information, strong linear edges, image noise, repetitive visual textures, distortions due to changes in the point of view, and obscurations due to scene discontinuities or moving objects.

Some of these conditions, such as low information, strong linear edges, and repetitive visual textures, can be detected in a single image. Matching of these areas can then be deferred until more global information (such as a set of reliable adjacent matches) is available to guide the matching process. Potential mismatches due to distortion, obscuration, or motion cannot be detected from a single image. Instead, the matching process or its results must be used to determine that a mismatch has occurred. Steps can then be taken to cope with the problem.

Traditional methods for detecting mismatches are based on the concept that correlation coefficient values will be low if an area is paired with anything other than its proper match. This condition has been detected by thresholding the correlation value—sometimes with an arbitrary threshold (0.5 is popular), sometimes with a threshold based on the expected correlation of random noise for a given window size, and sometimes with a threshold based on the information content of the window being matched [Hannah, 1974]. These methods can give false indications of mismatches the presence of noise, uncorrected distortion, scene discontinuities, and they can falsely ver an inappropriate match due to a moving object.

What is needed is a technique for evaluatin set of matches to determine which, if any, of t are inconsistent with the rest. Global techniqu such as fitting polynomials to the disparity d and evaluating each point on how well the polynomials fit it, have serious drawbacks. Such techniques apply the same criteria to all areas of a set of matches, despite the fact that terrain roughness, hence disparity differences, can vary greatly from one area to another. These variations can, for instance, cause a moderate error in a flat area to be missed while a large change in disparity at a mountain peak in a rough area is mistakenly marked as an error.

DISPARITY CONSTRAINTS

A more promising approach is the use of relaxation-like techniques for detecting local errors in the disparities by the application of constraints on the changes in the disparities. (Relaxation is an iterative technique for establishing the likeliness of a data point, depending on how well that point is supported by neighboring data points [Rosenfeld, et.al., 1976].) A similar approach has been developed for the detection of terrain elevation inconsistencies and has been used successfully to evaluate and rectify full grids of digital elevation data [Hannah, 1981]. To be useful in the application of assessing the reliability of a set of matches, the technique has been extended from grids of data points to arbitrarily spaced points, and from elevation data to two-dimensional disparity vectors. (The disparity approach, like its parent terrain elevation evaluation algorithm, assumes that the scene being evaluated is continuous and reasonably smooth, and that there are sufficient disparities to adequately sample the variations in the surface.)

In the grid-based elevation approach, each point had obvious neighbors—the horizontally, vertically, and diagonally adjacent points on the grid. In the general case, the concept of neighborhood implicit in the grid does not exist, so must be explicitly created as a pre-processing step. The matches to be evaluated are first paired, i.e. each point to be evaluated is associated with the N points which lie closest to it in the image. This process is constrained by first dividing the image into a grid of "buckets"; neighboring points can only come from the current or one of the eight adjacent buckets. Triples are formed at each point by combining that point with any two of its neighbors such that the angle formed by the three points is within a specified angle (usually 45 degrees) of 180 degrees, i.e., the three points are nearly co-linear. Only the M straightest triples are kept for each point. Note that a point may participate in up to M triples as the center point, plus up to N triples as one of the endpoints.

Each triple k has two values associated with it. One is an estimation of the reasonableness of that sequence of three disparities which is based on the vector second derivative of the disparities

$$\mathrm{DERIV2}(k) = \frac{\mathrm{DISP}(\mathrm{cent}(k)) - \mathrm{DISP}(\mathrm{end1}(k))}{\mathrm{DIST}(\mathrm{cent}(k),\mathrm{end1}(k))} - \frac{\mathrm{DISP}(\mathrm{end2}(k)) - \mathrm{DISP}(\mathrm{cent}(k))}{\mathrm{DIST}(\mathrm{cent}(k),\mathrm{end2}(k))}$$

where DISP is the disparity vector at a point of the triple, DIST is the distance between the center and an endpoint of the triple, and 'cent', 'end1', and 'end2' are the center and two endpoints of the triple, respectively. The components of this vector are compared to thresholds established by the user to form EST(k), the estimate of reasonableness for this triple. EST has values in the range of -1.0 to 1.0 (0.0 indicates a reasonable center disparity, 1.0 indicates a disparity that is too large, and -1.0 indicates a disparity that is too small). The second value associated with the triple is WT(k), a weight to be used with the reasonableness value; this number is based on the amount by which the three points of the triple deviate from a straight line, so that straight triples have more effect on the result than do bent triples.

The reliability R of each point is now established iteratively. The reliabilities are initially estimated to be the value of the correlation coefficient associated with each matched point. (We assume that all matches with negative correlations have been rejected!) This is iteratively refined by

$$R(i,n) = 1.0 - \mathrm{Abs}\left(\frac{\sum_k \mathrm{REL}(i,k,n-1) * \mathrm{EST}(k)}{\sum_k \mathrm{REL}(i,k,n-1)}\right)$$

That is, the reliability R of the i-th point on the n-th iteration is 1.0 minus the absolute value of the weighted average of the reasonableness estimates EST taken over all triples k involving the point i. The weighting factor REL is the product of the "straightness" weight WT associated with the triple and the minimum of the reliabilities R of the other two points in the triple on the previous iteration, i.e.

$$\mathrm{REL}(i,k,n-1) = \mathrm{WT}(k) * \mathrm{Minimum}(R(j1,n-1),\ R(j2,n-1))$$

$$j1,j2 \text{ in } k, \text{ but not } = i$$

Thus the reasonableness of a triple has an effect in proportion to the minimal reliability of the triple, preventing a bad match from prejudicing the evaluation of points around it.

The reliabilities are iterated until they appear to have converged, that is, until they change less than a user-specified threshold from one iteration to the next. In practice, it usually requires three iterations for the reliabilities to converge within a tolerance of 0.01.

RESULTS

Figure 1 shows the results of applying this algorithm. The underlying image data has been formed into a reduction hierarchy with reduction factor N = 2. The points to be matched were chosen by an "interest operator", which selects well-spaced areas having reasonable information content. Each point was then matched by a hierarchical matching technique [Hannah, 1980]. The figure shows the results of this matching by overlaying a symbol at the positions of the matching points in each image. In addition, each point in the left image has emanating from it the disparity vector associated with that match, in effect pointing to the position occupied by the matching point in the right image.

Examination of the disparity vectors reveals some obvious mismatches--most of the disparity vectors point strongly to the left and up slightly, but two near the top have significant downward components. The matches have been graded based on their reliability; the numeral which marks each match is the first digit of the reliability (i.e., matches marked with a 9 have a reliability greater than 0.9). The algorithm has correctly identified the two obvious mismatches near the top of the image (graded 0 and 1), along with a less obvious one in the lower left quarter of the image (graded 2). Two fairly subtle mismatches near the ridge line both received reasonably high grades of 7; all other matches are correct to within one pixel. (The correctness of the matches was established by comparison to results obtained by the U. S. Army Engineer Topographic Laboratories, Ft. Belvoir, VA, using an interactively "coached" correlation matching process. The author is indebted to ETL for this imagery and disparity data.)

CONCLUSIONS

This algorithm for the determination of the reliability of a set of match performs quite creditably in the domain for which it was designed. In this example, it correctly evaluated 49 out of 51 matches, missing only two subtle errors.

REFERENCES

Hannah, M. J., 1974. Computer Matching of Areas in Stereo Images, Ph.D. Thesis, AIM-239, Computer Science Department, Stanford University.

Hannah, M. J., 1980. "Bootstrap Stereo", Proceedings: Image Understanding Workshop, College Park, MD, April 30, 1980.

Hannah, M. J., 1981. "Error Detection and Correction in Digital Terrain Models", Photogrammetric Engineering and Remote Sensing, Vol. 47, No. 1, January, 1981, pp. 63-69.

Rosenfeld, A, R. A. Hummel, and S. W. Zucker, 1976. "Scene Labeling by Relaxation Operations", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-6, No. 6.

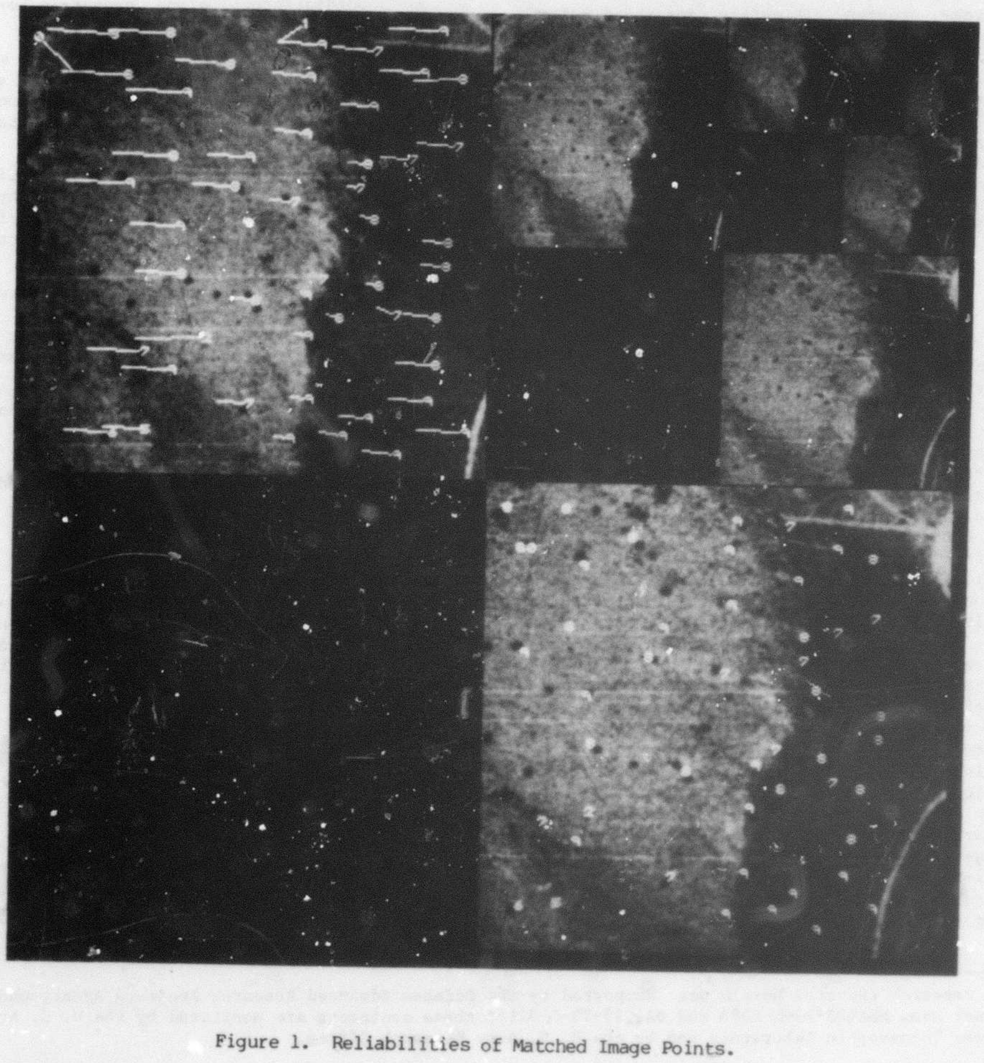


Figure 1. Reliabilities of Matched Image Points.

AD-P000 136

MODELING AND USING PHYSICAL CONSTRAINTS IN SCENE ANALYSIS*

M. A. Fischler, S. T. Barnard, R. C. Bolles, M. Lowry,
L. Quam**, G. Smith, and A. Witkin**

SRI International, Menlo Park, California 94025

ABSTRACT

This paper describes the results obtained in a research program ultimately concerned with deriving a physical sketch of a scene from one or more images. Our approach involves modeling physically meaningful information that can be used to constrain the interpretation process, as well as modeling the actual scene content. In particular, we address the problems of modeling the imaging process (camera and illumination), the scene geometry (edge classification and surface reconstruction), and elements of scene content (material composition and skyline delineation).

I INTRODUCTION

Images are inherently ambiguous representations of the scenes they depict: images are 2-D views of 3-D space, they are single slices in time of ongoing physical and semantic processes, and the light waves from which the images are constructed convey limited information about the surfaces from which these waves are reflected. Therefore, interpretation cannot be strictly based on information contained in the image; it must involve, additionally, some combination of a priori models, constraints, and assumptions. In current machine-vision systems this additional information is usually not made explicit as part of the machine's data base, but rather resides in the human operator who choses the particular techniques and parameter settings to reflect his understanding of the scene context. This paper describes a portion of the SRI program in machine vision research that is concerned with identifying and modeling physically meaningful information that can be used to automatically constrain the interpretation process. In particular, as an adjunct to any autonomous system with a generalized competence to analyze imaged data of 3-D real-world scenes, we believe that it is necessary to explicitly model and use the following types of knowledge:

(1) Camera model and geometric constraints (location and orientation in space from which the image was acquired, vanishing points, ground plane, geometric horizon, geometric distortion).

(2) Photometric and illumination models (atmospheric and image-processing system intensity-transfer functions, location and spectrum of sources of illumination, shadows, highlights).

(3) Physical surface models (description of the 3-D geometry and physical characteristics of the visible surfaces; e.g., orientation, depth, reflectance, material composition).

(4) Edge classification (physical nature of detected edges; e.g., occlusion edge, shadow edge, surface intersection edge, material boundary edge, surface marking edge).

(5) Delineation of the visible horizon (skyline)

(6) Semantic context (e.g., urban or rural scene, presence of roads, buildings, forests, mountains, clouds, large water bodies, etc.).

In the remainder of this paper, we will describe in greater detail the nature of the above models, our research results concerning how the parameters for some of these models can be automatically derived from image data, and how the models can be used to constrain the interpretation process in such tasks as stereo compilation and image matching.

If we categorize constraints according to the scope of their influence, then the work we describe is primarily concerned with global and extended constraints rather than with constraints having only a local influence. To the extent that constraints can be categorized as geometric, photometric, or semantic and scene dependent, it would appear that we have made the most progress in understanding and modeling the geometric constraints.

* The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract Nos. MDA903-79-C-0588 and DAAG29-79-C-0216; these contracts are monitored by the U. S. Army Engineer Topographic Laboratory and by the U. S. Army Research Office.

** Current affiliation: Fairchild Artificial Intelligence Laboratory, Palo Alto, California.

II CAMERA MODELS AND GEOMETRIC CONSTRAINTS

The camera model describes the relationship between the imaging device and the scene; e.g., where the camera is in the scene, where it is looking, and more specifically, the precise mapping from points in the scene to points in the image. In attempting to match two views of the same scene taken from different locations in space, the camera model provides essential information needed to contend with the projective differences between the resulting images.

In the case of stereo reconstruction, where depth (the distance from the camera to a point in the scene) is determined by finding the corresponding scene point in the two images and using triangulation, the camera models (or more precisely, the relative camera model) limit the search for corresponding points to one dimension in the image via the "epipolar" constraint. The plane passing through a given scene point and the two lens centers intersects the two image planes along straight lines; thus a point in one image must lie along the corresponding (epipolar) line in the second image, and one need only search along this line, rather than the whole image to find a match.

When human interaction is permissible, the camera model can be found by having the human identify a number of corresponding points in the two images and using a least-squares technique to solve for the parameters of the model [5]. If finding the corresponding points must be carried out without human intervention, then the differences in appearance of local features from the two viewpoints will cause a significant percentage of false matches to be made; under these conditions, least squares is not a reliable method for model fitting. Our approach to this problem [3] is based on a philosophy directly opposite to that of least-squares -- rather than using the full collection of matches in an attempt to "average out" errors in the model-fitting process, we randomly select the smallest number of points needed to solve for the camera model and then enlarge this set with additional correspondences that are compatible with the derived model. If the size of the enlarged compatibility set is greater than a bound determined by simple statistical arguments, the resulting point set is passed to a least-squares routine for a more precise solution. We have been able to show that as few as three correspondences are sufficient to directly solve for the camera parameters when the three-space relationships of the corresponding points are known; a recent result [13] indicates that 5 to 8 points are necessary to solve for the relative camera model parameters when three space information is not available a priori.

The perspective imaging process (the formation of images by lenses) introduces global constraints that are independent of the explicit availability of a camera model; particularly important are the detection and use of "vanishing points." A set of parallel lines in 3-D space, such as the vertical edges of buildings in an urban scene, will project onto the image plane as a set of straight lines intersecting at a common point. Thus, for example, if we can locate the vertical vanishing point, we can strongly constrain the search for vertical objects such as telephone or power poles or building edges, and we can also verify conjectures about the 3-D geometric configuration of objects with straight edges by observing which vanishing points these edges pass through. The two horizontal vanishing points corresponding to the rectangular layout of urban areas, the vanishing point associated with a point of illumination [8], and the vanishing point of shadow edges projected onto a plane surface in the scene, provide additional constraints with special semantic significance. The detection of clusters of straight parallel lines by finding their vanishing points can also be used to automatically screen large amounts of imagery for the presence of man-made structures.

The technique we have employed to detect potential vanishing points involves local edge detection by finding zero-crossings in the image convolved with both Gaussian and Laplacian operators [9], fitting straight line segments to the closed zero-crossing contours, and then finding clusters of intersection points of these straight lines. In order to avoid the combinatorial problem of computing intersection points for all pairs of lines, or the even more unreasonable approach of plotting the infinite extension of all detected line segments and noting those locations where they cluster, we have implemented the following technique. Consider a unit radius sphere physically positioned in space somewhere over the image plane (there are certain advantages to locating the center of the sphere at the camera focal point if this is known, in which case it becomes the Gaussian sphere [6,7], but any location is acceptable for the purpose under consideration here). Each line segment in the image plane and the center of the sphere define a plane that intersects the sphere in a great circle -- if two or more straight lines intersect at the same point on the image plane, their great circles will intersect at two common points on the surface of the sphere, and the line passing through the center of the sphere and the two intersection points on the surface of the sphere will also pass through the intersection point in the image plane.

III EDGE CLASSIFICATION

An intensity discontinuity in an image can correspond to many different physical events in the scene, some very significant for a particular purpose, and some merely confusing artifacts. For example, in matching two images taken under different lighting conditions, we would not want to use shadow edges as features; on the other hand, shadow edges are very important cues in looking for (say) thin raised objects. In stereo matching, occlusion edges are boundaries that area correlation patches should not cross (there will also be a region on the "far" side of an occlusion edge in which no matches can be found); occlusion edges also define a natural distance progression in an image even in the absence of stereo information. If it is possible to assign labels to detected edges describing their physical nature, then those interpretation processes that use them can be made much more robust.

We have implemented an approach to detecting and identifying both shadow and occlusion edges, based on the following general assumptions about images of real scenes:

(1) The major portion of the area in an image (at some reasonable resolution for interpretation) represents continuous surfaces.

(2) Spatially separated parts of a scene are independent, and their image projections are therefore uncorrelated.

(3) Nature does not conspire to fool us; if some systematic effect is observed that we normally would anticipate as caused by an expected phenomena due to imaging or lighting, then it is likely that our expectations provide the correct explanation; e.g., coherence in the image reflects real coherence in the scene, rather than a coincidence of the structure and alignment of distinct scene constituents.

Consider a curve overlayed on an image as representing the location of a potential occlusion edge in the scene. If we construct a series of curves parallel to the given one, then we would expect that for an occlusion edge, there would be a high correlation between adjacent curves on both sides of the given curve, but not across this curve. That is, on each side, the surface continuity assumption should produce the required correlation, but across the reference curve the assumption of remote parts of the scene being independent should produce a low correlation score. In a case where the reference curve overlays a shadow edge, we would expect a continuous high (normalized) correlation between adjacent curves on both sides and across the reference curve, but the regression coefficients should show a discontinuity as we cross the reference curve. This technique is described in greater detail in [14]. Figures 1 and 2 show experimental results for shadow and occlusion edges.

IV INTENSITY MODELING (& Material Classification)

Given that there is a reasonably consistent transform between surface reflectance and image intensity, the exact nature of this transform is not required to recover rather extensive information about the geometric configuration of the scene. It is even reasonable to assume that shadows and highlights can be detected without more precise knowledge of the intensity mapping from surface to image; but if we wish to recover information about actual surface reflectance or physical composition of the scene, then the problem of intensity modeling must be addressed.

Even relatively simple intensity modeling must address three issues: (1) the relationship between the incident and reflected light from the surface of an object in the scene as a function of the material composition and orientation of the surface; (2) the light that reaches the camera lens from sources other than the surface being viewed (e.g., light reflected from the atmosphere); and (3) the relationship between the light reaching the film surface and the intensity value ultimately recorded in the digital image array.

Our approach to intensity modeling assumes that we have no scene-specific information available to us other than the image data. We use a model of the imaging process that incorporates our knowledge of the behavior of the recording medium, the properties of atmospheric transmission, and the reflective properties of the scene materials. For aerial imagery we use an atmospheric model that assumes a constant amount of light, (independent of scene radiance), is scattered by the atmosphere into the camera.

$$I=R+S$$

where I is the image irradiance, R the scene radiance, and S the image irradiance caused by atmospheric scattering. We use a logarithmic relationship between image irradiance and film density D,

$$D= a*\log(I) + d$$

where a and d are film constants, whose values need to be calculated. For a surface radiance model we assume Lambertian behaviour (the reflected light is proportional to the incident light, the constant of proportionality is a function of the surface material, and the relative brightness of the surface is independent of the location of the viewer).

$$R=EAN$$

where E is the illumination strength (scene irradiance), A the surface reflection or albedo, and N a function related to the effects of surface orientation (for Lambertian surfaces this is a function of the angle between the surface normal and the light direction).

If for the present we ignore surface orientation effects, that is we assume all surfaces are orientated in the same direction, then our model has the form

$$D=a*\log(A+b)+c$$

where a,b,and c are constants that need to be determined. b is the ratio of atmospheric scattering to illumination irradiance.

We calibrate our model by identifying a few regions of known material in an image. Three materials are sufficient. The fitting is achieved by guessing b - we know b lies in the range 0 to 1 - applying the least squares method to the resultant linear equation to calculate a,c, and the residual sum, and adjusting b to minimize this residual sum.

The resultant model is used to transform the given image into a new image depicting the scene albedo. The albedo image has been used to provide an initial classification (and partitioning) of the scene using straight forward classification techniques based on "known" surface albedos. This technique allows classification without the need to provide training samples of all classes that are present in the image.

V SHADOW DETECTION (and Raised Object Cueing)

The ability to detect and properly identify shadows is a major asset in scene analysis. For certain types of features, such as thin raised objects in a vertical aerial image, it is often the case that only the shadow is visible. Knowledge of the sun's location and shadow dimensions frequently allows us to recover geometric information about the 3-D structure of the objects casting the shadows, even in the absence of stereo data [8,10]; but perhaps just as important, distinguishing shadows from other intensity variations eliminates a major source of confusion in the interpretation process.

Given an intensity discontinuity in an image, we can employ the edge labeling technique described earlier to determine if it is a shadow edge. However, some thin shadow edges are difficult to find, and if there are lots of edges, we might not want to have to test all of them to locate the shadows. We have developed a number of techniques for locating shadow edges directly, and will now describe a simple but effective method for finding the shadows cast by thin raised objects (and thus locating the objects as well).

We assume we either know the approximate sun direction, or equivalently, the shadow vanishing point. We first employ a thin line detector oriented parallel to the sun direction at every location in the image, and then apply a moving-window averaging technique in the sun's direction to further enhance the line detector's response and reduce noise. The result of these operations is to smear both the noise and the thin shadow lines. We next thin the shadow lines, eliminate all weak responses, and overlay the result on the original image. T'.e foot of each shadow line now points to the base of the thin raised object casting the shadow. Given the results from two (or more) images taken at different times, the intersections of shadow lines locates the objects more precisely and also eliminates false alarms.

The same technique has been applied to the detection of raised objects of extended size. Shadow edges of the extended object are detected and used to locate the object. Figures 3-10 show this approach to detecting both thin and extended raised objects.

VI VISUAL SKYLINE DELINEATION

Although not always a well defined problem, delineation of the land-sky boundary provides important constraining information for further analysis of the image. Its very existence in an image tells us something about the location of the camera relative to the scene (i.e., that the scene is being viewed at a high-oblique angle), allows us to estimate visibility (i.e., how far we can see -- both as a function of atmospheric viewing conditions, and as a function of the scene content), provides a source of good landmarks for (autonomous) navigation, and defines the boundary beyond which the image no longer depicts portions of the scene having fixed geometric structure.

In our analysis, we generally assume that we have a single right-side-up image in which a (remote) skyline is present. Confusing factors include clouds, haze, snow-covered land structures, close-in raised objects, and bright buildings or rocks that have intensity values identical to those of the sky (a casual inspection of an image will often provide a misleading opinion about the difficulty of skyline delineation for the given case). Our initial approach to this problem was to investigate the use of sli\ghtly modified methods for linear delineation [4] and histogram partitioning based on intensity and texture measures; we employ fairly simple models of the relationship between land, sky, and cloud brightness and texture.

Currently, we are employing a region based technique which operates as follows:

To eliminate spurious regions and gaps in region boundaries caused by noise we first reduce the given image by a factor of at least 4. We partition the image into a nested pyramid of regions; each region being one in which every pixel has an intensity value which differs by less than some given threshold, from a least one of its 4-neighbors. The nested pyramid is constructed by using a sequence of increasing threshold values (e.g. 2,4,8,...); thus if T1 and T2 are thresholds such that T1 < T2, then any region found with threshold T1 is necessarily identical to a subregion or a region found with threshold T2.

A "sky seed" is found by identifying the region that dominates the very top of the picture with a segmentation threshold of 2 (this is the lowest threshold that allows a gradient to exist within a region). "or a clear sky, or a sky with cumulus clouds comp.etely surrounded by clear sky, this step usually identifies the entire sky. Figure 11 shows an urban scene with overcast sky and figure 12 shows the same scene with the sky seed overlayed.

As an additional piece of information, the sky seed is classified as clear sky, overcast sky, or patchy clouds. Patchy cumulus clouds appear as large bright regions within a clear sky region, while the brightness function for a clear sky can be modeled as a linear function of the image coordinates. Although the equations governing clear sky luminance are complex integrodifferential equations, it was determined empirically that for the viewing angles produced by a 50mm lens, a (linear) planar model provided a good fit. To determine whether the sky seed is clear sky or overcast sky, a least squares fit to the planer model is made, and the mean square error, corrected by the measured intensity variance, is compared to a fixed thresholded. The classification of the sky into clear/overcast/patchy clouds can help to resolve some of the confusing factors in skyline detection, but this information is not currently used.

Next, a line spanning the picture from right to left is found that is either at or below the true skyline; this line is found by doubling the threshold for segmentation until the region containing the sky seed touches the bottom of the picture. Since we make an initial assumption that the sky does not touch the bottom of the picture, this threshold is then backed off by a factor of 2 and a "land seed" is defined as the complement of the sky region. The assumption here is that the skyline, or some boundary in the land region, is of higher contrast than any extended boundary within the sky. For all 15 test pictures that we employed in our experiments, this assumption was only violated once (where a particularily bright cumulus cloud on the horizon formed a brighter boundary with the sky than a bright rock on the horizon; such a situation can be easily detected after initial processing). Figure 13 shows the case in which a region containing the sky seed touches the bottom of the picture at a threshold of 16, and Figure 14 shows the picture split into a sky seed, land seed, and ambiguous unclassified portion. The land seed is determined by using a threshold of 8. Figure 15 shows an additional and more typical example of skyline delineation

In a substantial number of pictures the sky and land seeds touch, thereby delineating the skyline. If the sky and land seeds do not have a common boundary, a portion of the picture is left unclassified, bounded by the sky seed above and the land region below. Current work focuses on developing methods to disambiguate the unclassified portion of the picture. The methods under development are generic to all types of scenes and our approach does not use semantic knowledge of particular land features. Prior work on this topic, employing considerable semantic knowledge, is contained in Sloan [11].

VII SURFACE MODELING

Obtaining a detailed representation of the visible surfaces of the scene, as (say) a set of point arrays depicting surface orientation, depth, reflectance, material composition, etc., is possible from even a single black and white image [12,2]. A large body of work now exists on this topic,(see [15,16] for recent work by our group), and although directly relevant to our efforts, it is not practical to attempt a discussion of this material here. There is, however, one key difference between surface modeling and the other topics we have discussed -- the extent to which the particular physical knowledge modeled constrains the analysis of other parts of the scene. In this paper we have been primarily concerned with physical models that provide global or extended constraints on the analysis; surface modeling via point arrays provides a very localized constraining influence.

VIII CONSTRAINT-BASED STEREO COMPILATION

The computational stereo paradigm encompasses many of the important task domains currently being addressed by the machine-vision research community [1]; it is also the key to an application area of significant commercial and military importance -- automated stereo compilation. Conventional approaches to stereo compilation, based on finding dense matches in a stereo image pair by area correlation, fail to provide acceptable performance in the presence of the following conditions typically encountered in mapping cultural or urban sites: widely separated views (in space or time), wide angle views, oblique views, occlusions, featureless areas, repeated or periodic structures. As an integrative focus for our research, and because of its potential to deal with the factors that cause failure in the conventional approach, we are constructing a constraint-based stereo system that encompasses many of the physical modeling techniques discussed above.

Figure 16 show how a stereo system can exploit global geometric constraints. First, straight lines and vanishing points are found in the two stereo images as described earlier (see Section II). Lines are first classified according to which vanishing point they pass through. Those lines not associated with the detected vanishing points are ignored. The vanishing points in the two stereo views are then matched. The direction in space established by a vanishing point is a feature of the scene which is invariant under translation of the camera. Two matches of vanishing points are sufficient to calculate the rotational differences between the cameras i.e., the rotation required to bring one camera's vanishing points into congruence with the other's. Two matches of ordinary points are now sufficient to determine the translation of one camera with respect to the other (up to an unknown scaling factor).

Using vanishing points can improve stereo matching even when the exact camera model is unknown. In Figures 16i and 16j, lines passing through a vanishing point in one image are first matched to the set of lines passing through the corresponding vanishing point in the other image. For example, right-image lines passing through the vertical vanishing point are only matched to left-image lines that also pass through the vertical vanishing point. Within these subsets, lines are matched according to a score based on four features: (1) difference in distance from the vanishing point to the lines, (2) ratio of lengths, (3) difference in contrast and, (4) difference in phase i.e., the angle the line makes with the image-horizontal. Each subscore is a value in the interval [0,1]. The value represents the likelihood of this combination of the four features. The subscores are combined multiplicatively, and the combination with the maximum score (above a preset threshold) is chosen. Even this simple matching technique, using no search or relaxation, finds an adequate number of correct matches.

IX CONCLUDING COMMENTS

When a person views a scene, he has an appreciation of where he is relative to the scene, which way is up, the general geometric configuration of the surfaces (especially the support and barrier surfaces), and the overall semantic context of the scene. The research effort we have described is intended to provide similar information to constrain the more detailed interpretation requirements of machine vision (e.g., such tasks as stereo compilation and image matching).

REFERENCES

1. S. T. Barnard and M. A. Fischler, "Computational Stereo," ACM Surveys, 1982 (in press).

2. H. G. Barrow and J. M. Tenenbaum, "Recovering intrinsic scene characteristics from images," in Computer Vision Systems (A. Hanson and E. Riseman ed.) Academic Press, pp. 3-26 (1978).

3. M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography," CACM, Vol. 24(6), pp. 381-395 (June 1981).

4. M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf, "Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique," Computer Graphics and Image Processing, Vol. 15(3), pp. 201-223 (March 1981).

5. D. B. Gennery, "Least-squares stereo-camera calibration," Stanford Artificial Intelligence Project Internal Memo, Stanford University (1975).

6. D. A. Huffman, "Realizable configuration of lines in pictures of polyhedra," in Machine Intelligence (Elcock and Michie, ed.), Edinburgh University Press, Edinburgh, Scotland, pp. 493-509 (1977).

7. J. R. Kender, "Shape from Texture," (Ph.D. Thesis, Report No. CMU-CS-81-102) Carnegie-Mellon University, Pittsburgh, Pennsylvania (November 1980).

8. D. G. Lowe and T. O. Binford, "The interpretation of three-dimensional structure from image curves," IJCAI-81, pp. 613-618 (1981).

9. D. Marr and E. C. Hildreth, "Theory of edge detection," MIT AI Lab Memo 518 (1979).

10. S. Shafer and T. Kanade, "Using shadows in finding surface orientations," (Report No. CMU-CS-82-100) Carnegie-Mellon University, Pittsburgh, Pennsylvania (January 1982).

11. K. R. Sloan, Jr., "World model driven recognition of natural scenes," University of Pennslyvania, Philadelphia, Pennsylvania (June 1977).

12. J. M. Tenenbaum, M. A. Fischler, and H. G. Barrow, "Scene Modeling: A structural basis for image description," Computer Graphics and Image Processing," Vol. 12(4), pp. 407-425 (April 1980).

13. R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," University of Illinois, Urbana, Illinois (August 1981).

14. A. Witkin, "Recovering intrinsic scene characteristics from images," SRI Project 1019, Interim Technical Report, SRI International, Menlo Park, California (September 1981).

15. A. P. Pentland, "Local computation of shape," Proceddings AAAI, (1982)

16. G. B. Smith, "The recovery of surface orientation from image irradiance," Image Understanding Workshop, Stanford (September 1982) (this volume)

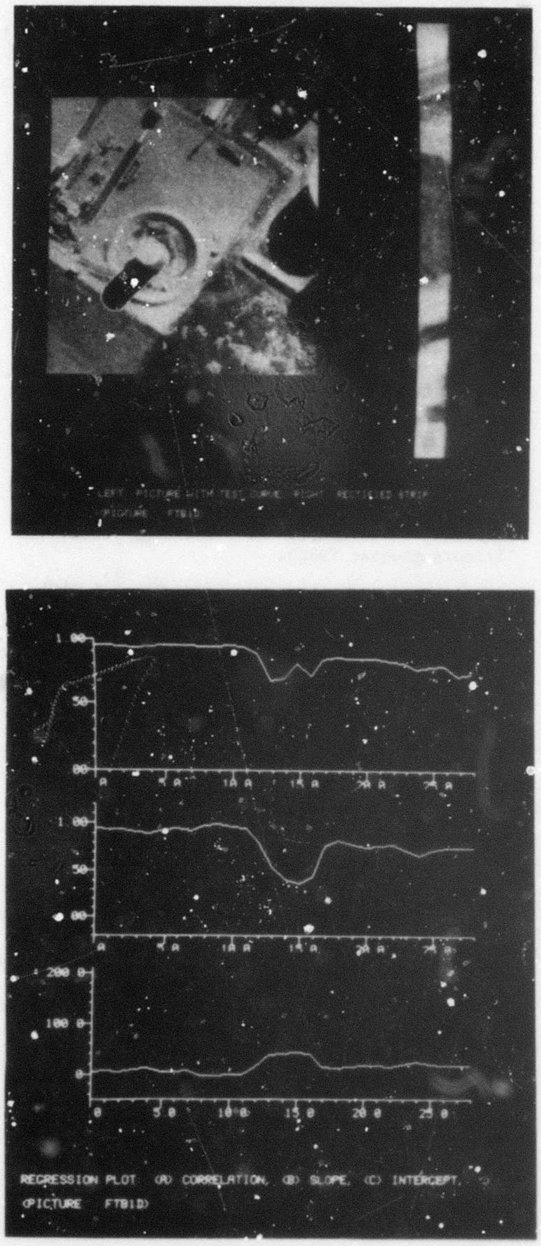


FIGURE 1 EXAMPLE OF CAST-SHADOW EDGE

(EDGE CLASSIFICATION)

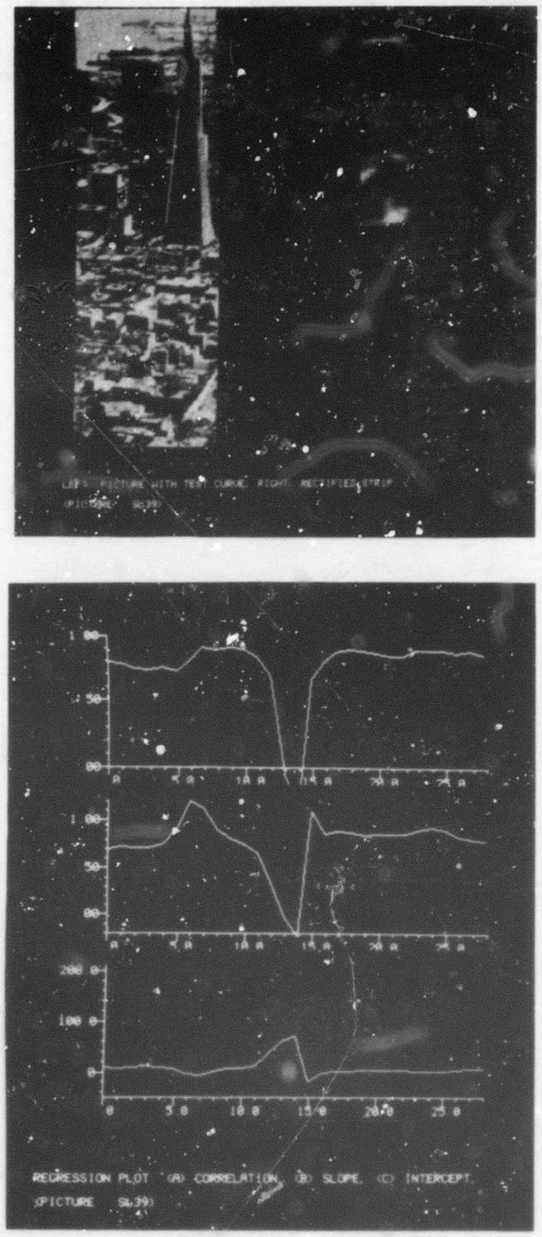
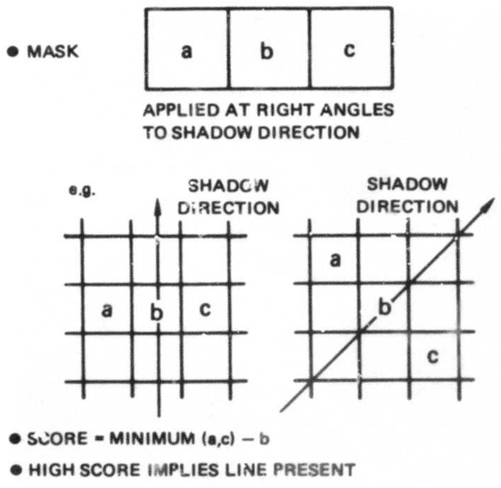


FIGURE 2 EXAMPLE OF EXTREMAL EDGE

(EDGE CLASSIFICATION)



● MASK
a
b
c
APPLIED AT RIGHT ANGLES TO SHADOW DIRECTION
e.g.
SHADOW DIRECTION
a
b
c
SHADOW DIRECTION
a
b
c
● SCORE = MINIMUM (a,c) – b
● HIGH SCORE IMPLIES LINE PRESENT

FIGURE 3 THIN SHADOW LINE DETECTOR

(SHADOW DETECTION)

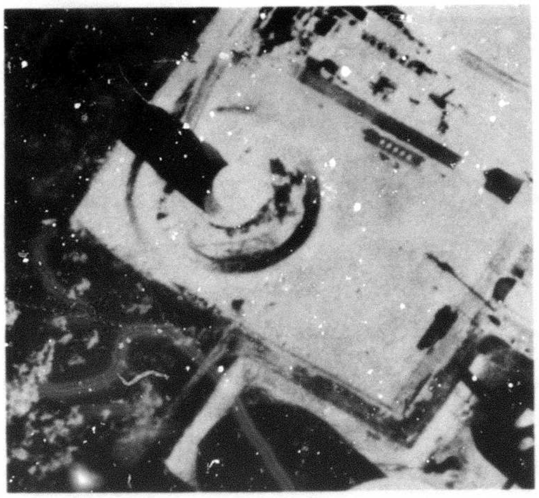


FIGURE 4 ORIGINAL IMAGE

(SHADOW DETECTION)

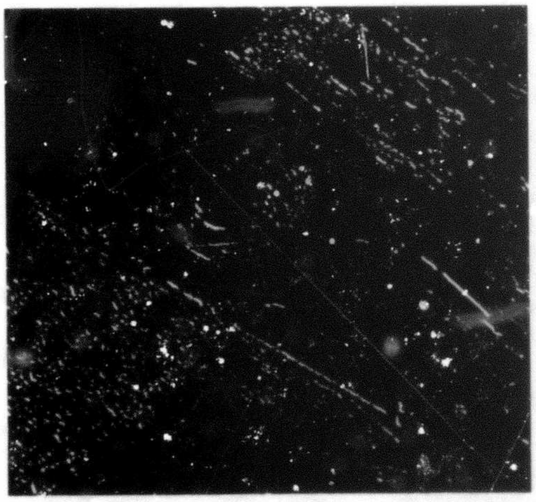


FIGURE 5 RESULTS OF APPLYING THE LINE DETECTOR TO ORIGINAL IMAGE

(SHADOW DETECTION)

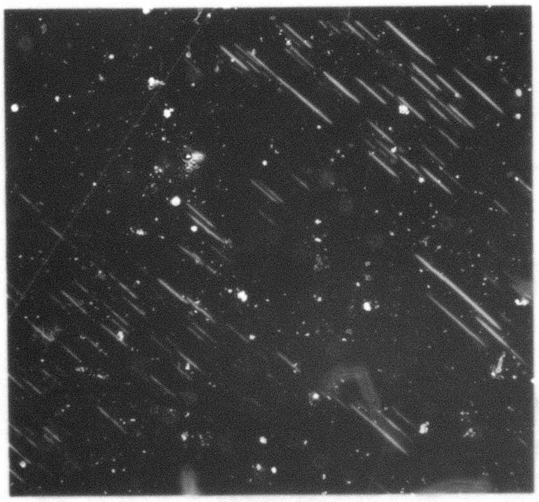


FIGURE 6 NOISE REMOVAL USING MOVING WINDOW INTEGRATION ALONG SHADOW DIRECTION

(SHADOW DETECTION)

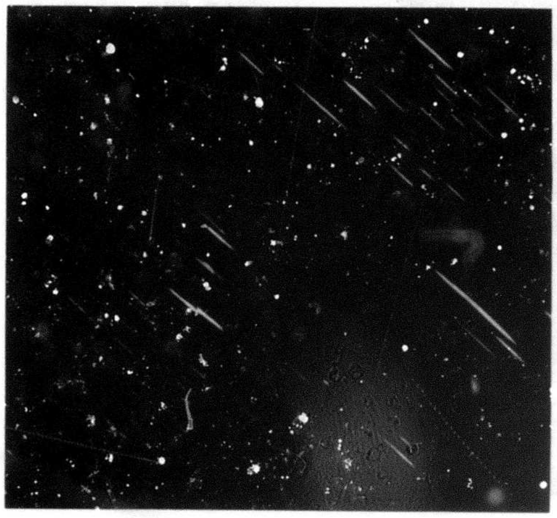


FIGURE 7 LINE THINNING
(SHADOW DETECTION)

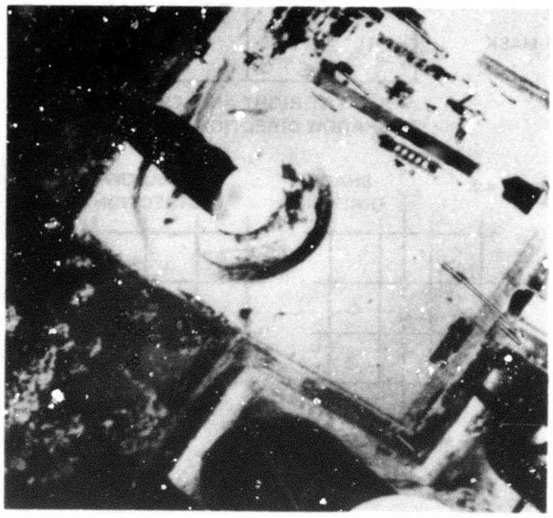


FIGURE 8 THRESHOLDED LINES OVERLAYED ON ORIGINAL IMAGE: COMPARE WITH FIGURE 4
(SHADOW DETECTION)

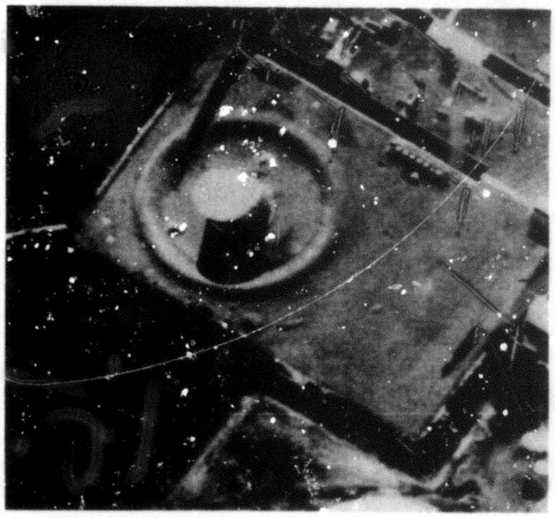
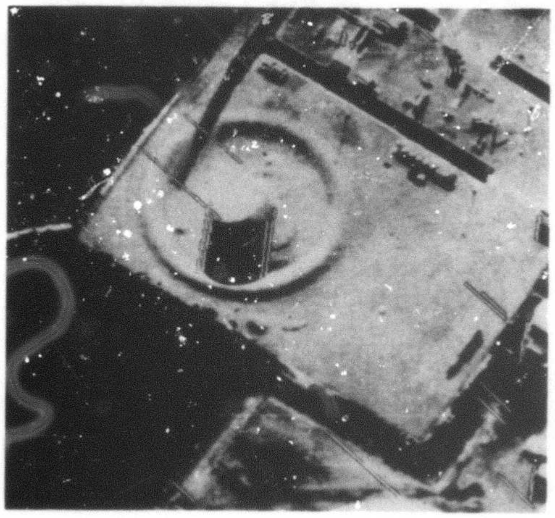


FIGURE 9 RESULTS USING TWO ADDITIONAL IMAGES: COMPARE WITH FIGURE 10
(SHADOW DETECTION)



FIUGRE 10 RESULTS FOR DETECTING EXTENDED OBJECTS
(SHADOW DETECTION)

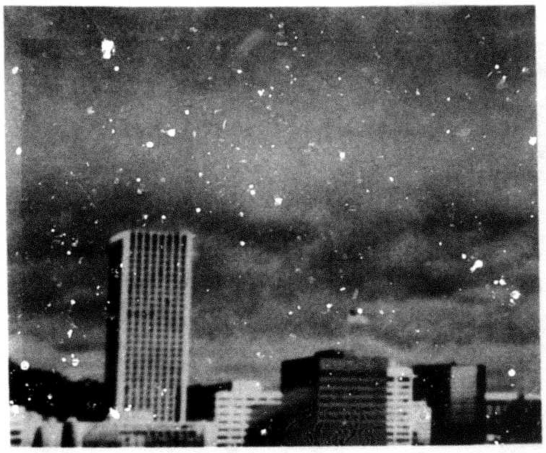


FIGURE 11 URBAN SCENE WITH OVERCAST SKY

(SKYLINE DELINEATION)

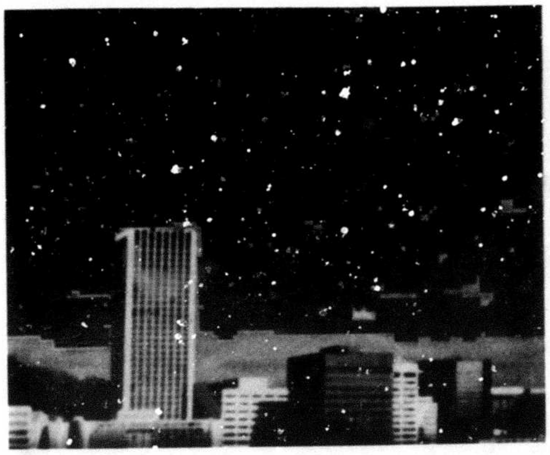


FIGURE 12 SKY SEED FOUND WITH REGION SEGMENTATION AT THRESHOLD 2

(SKYLINE DELINEATION)

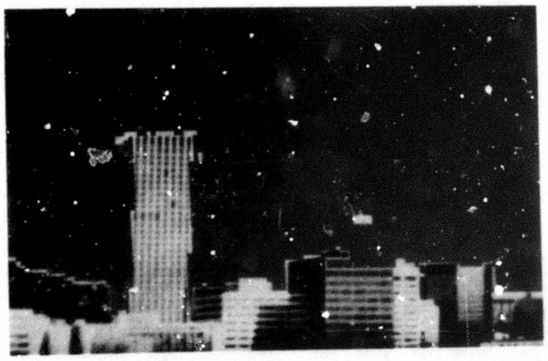


FIGURE 13 THRESHOLD IS DOUBLED UNTIL REGION CONTAINING SKY SEED TOUCHES "BOTTOM" 15% OF PICTURE

(SKYLINE DELINEATION)

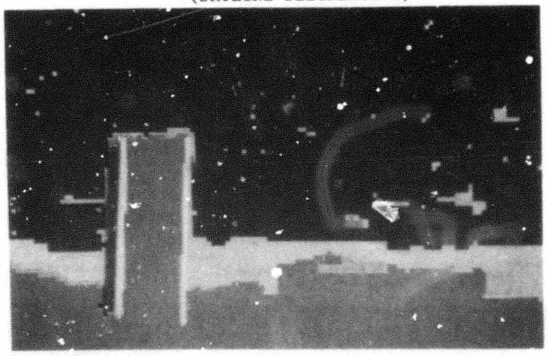
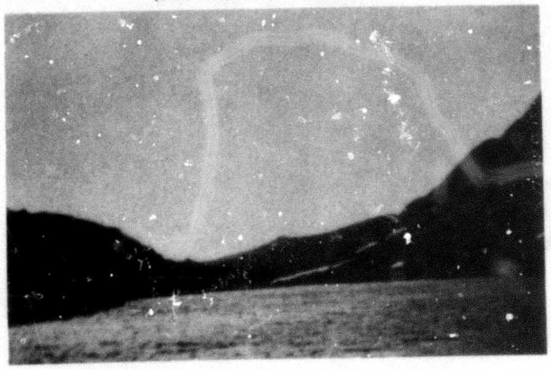
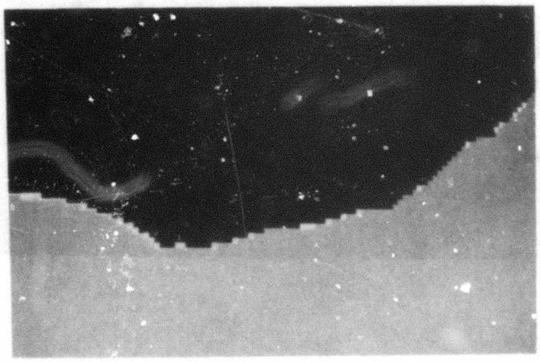


FIGURE 14 PICTURE SEGMENTED INTO SKY SEED, UNCLASSIFIED PORTION, AND LAND SEED

(SKYLINE DELINEATION)

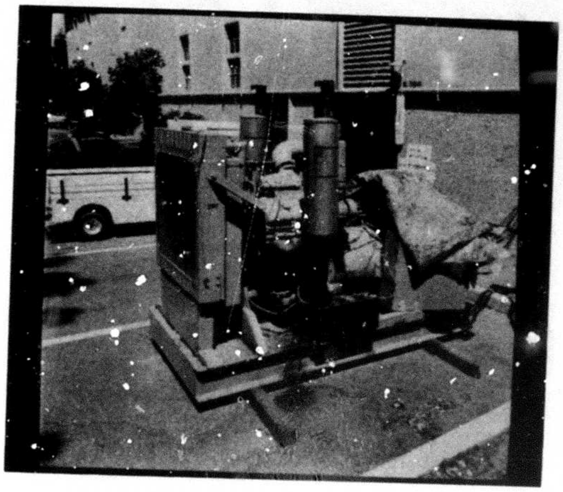


(a) Original Image



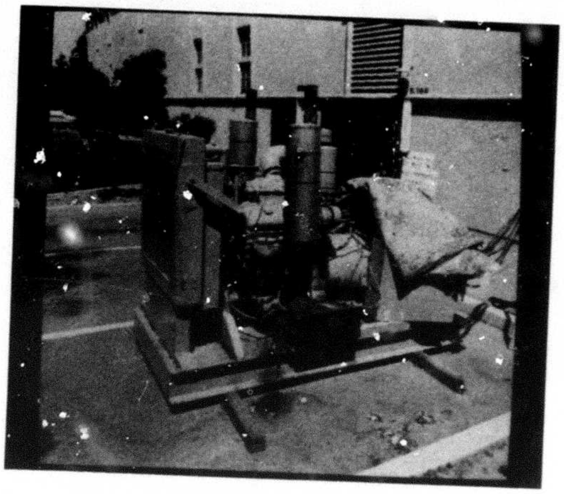
(b) Sky and land seed boundaries coincide at skyline to produce unambiguous delineation

FIGURE 15 SKYLINE DELINEATION

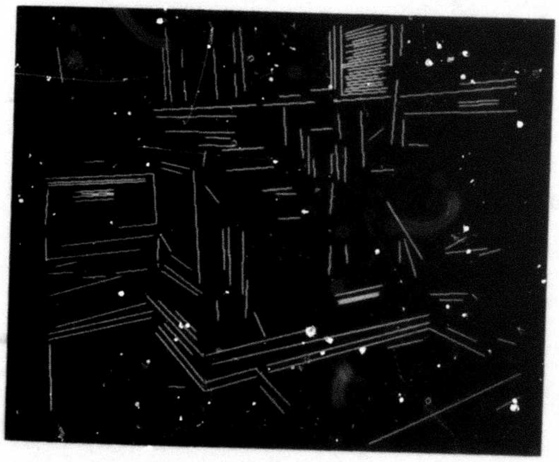


(a)

A Stereo Pair

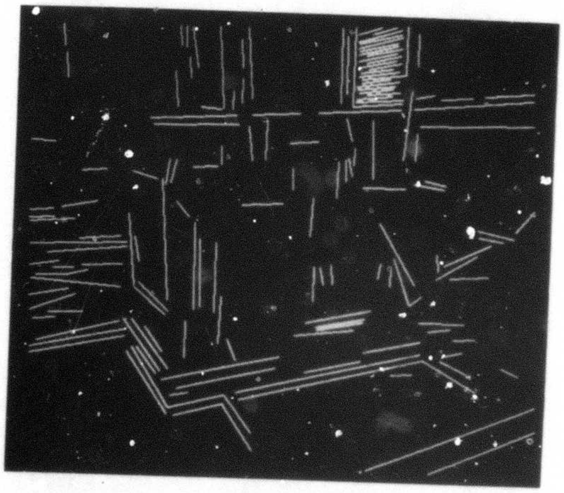


(b)



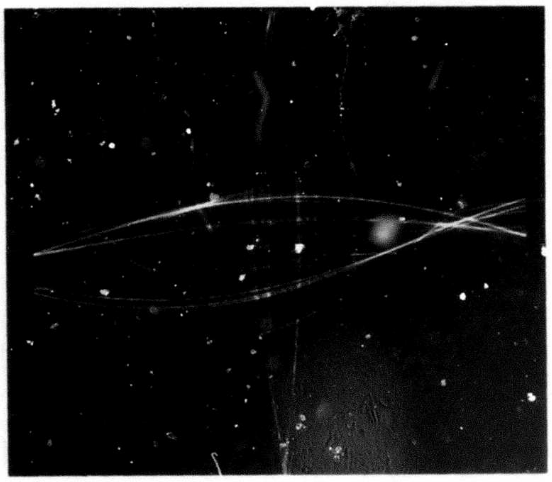
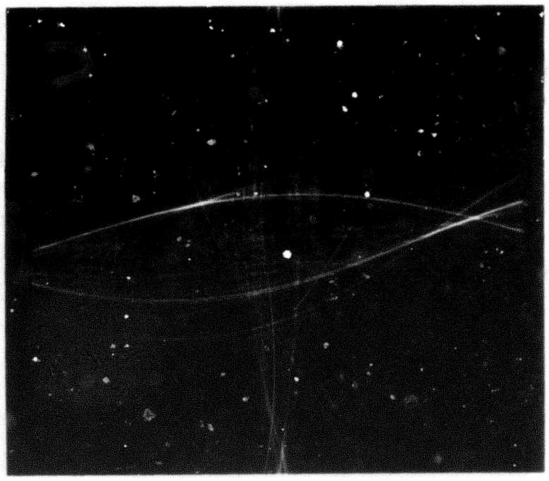
(c)

Line Segments



(d)

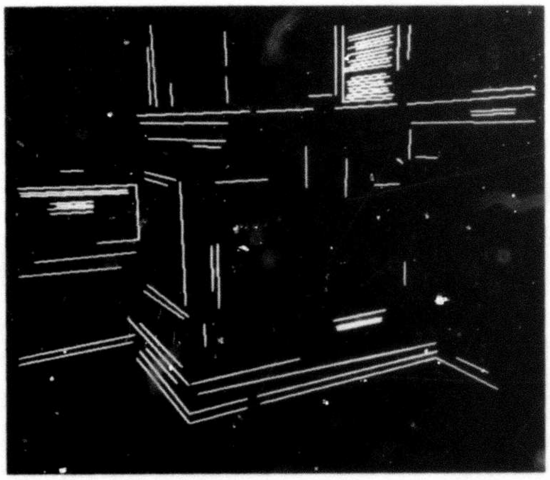
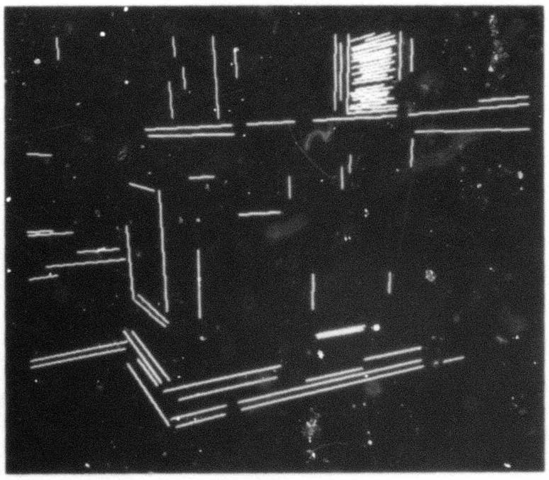
FIGURE 16 STEREO MATCHING USING GLOBAL PERSPECTIVE CONSTRAINTS

(e)

(f)

Gaussian Mapping and Vanishing Points

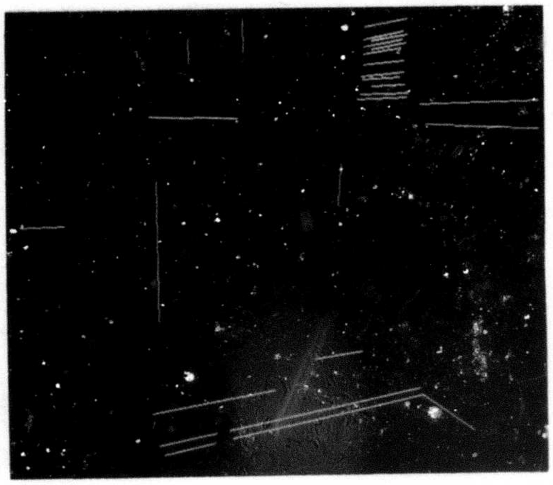
(g)

(h)

Parallel Lines

FIGURE 16 (CONTINUED)

STEREO MATCHING USING GLOBAL PERSPECTIVE CONSTRAINTS


(i)

Matched Lines

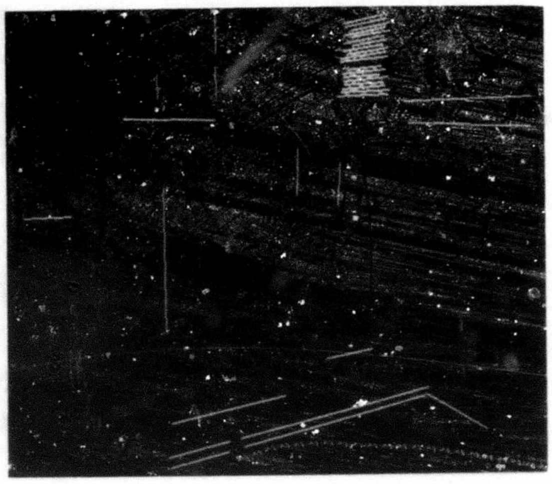

(j)

FIGURE 16 (CONTINUED)

STEREO MATCHING USING GLOBAL PERSPECTIVE CONSTRAINTS

AD-P000 137

SYMBOLIC MATCHING OF IMAGES AND SCENE MODELS

Keith E. Price
Image Processing Institute
University of Southern California
Los Angeles, California 90089-0272

Abstract

In this paper we explore the application of a general relaxation based matching procedure to the problem of matching pairs of images and the extension of basic techniques to matching hierarchical descriptions of scenes. These problems require extension to the general graph matching procedure which account for the special properties of the given task. The efficiency of the matching program is greatly improved by including these changes.

I. Introduction

Matching of images and descriptions has many different uses and can be performed at several different levels. Some matching tasks require that very precise corresponding locations are computed (*e.g., stereo depth computation, pixels level change* detection). But for many tasks, matching at a grosser level (i.e., finding correspondences between large areas) is best. This paper discusses results of a general symbolic level image matching system applied to the task of matching an image and an a priori description of the scene (a model) to locate objects in the image and the task of matching two images to find the location of an object in two different views. Thus we use this program to find correspondences between areas of the images (or objects) rather than to find a pixel level mapping between them.

II. Background

The work reported here represents an extension of earlier relaxation based symbolic matching efforts [1]. A variety of image matching techniques have been developed for different tasks. Moravec [2] has developed a system which locates feature points in one image (essentially corners) and uses a correlation based matching procedure at multiple resolutions to efficiently find a set of corresponding points in the two images. This system is intended for land based robot navigation which uses the three dimensional information from these feature points for navigation. A stereo system developed by Baker [3] generates a complete disparity map starting from edge correspondences which can be used for depth computations if the camera positions are known. These two (and many other similar efforts) concentrate on precise matching of image data.

Several systems which work on a variety of symbolic representations have also been developed. Barnard and Thompson [4] have developed a relaxation based motion analysis program which finds corresponding feature points in two images. The feature points are similar to those of Moravec [2], but they are located in both images. Wong et al. [5] also use a relaxation procedure to match corners which are detected in pairs of images. This system allows arbitrary translations and rotations of the camera. Clark et al. [6] have developed a system to match line like structures (generally edges or region boundaries). The program uses three initial matching lines to get a mapping between the two images. The quality of the match depends on how well all the other lines match, and the best match is determined by trying all possible triples of matching lines. The number of possible triples is limited by the allowable transformations, i.e., given one match, the possible matches for the other two are very restricted. Gennery [7] extracts objects and uses a tree searching procedure to find the best match.

The relaxation procedure used here is developed more fully in [1, 8] and differs from other methods in its gradient optimization approach. There are several alternative relaxation updating schemes such as the basic method of Rosenfeld et al. [9], Peleg [10], and Hummel and Zucker [11]. We have implemented these other methods and use them for a comparison of the results.

III. Description and Matching

This matching system uses feature-based symbolic descriptions for its input. The description of an idealized version of the scene (a model) is developed by the user through an interactive procedure. The image descriptions are derived automatically from input images. The underlying descriptive mechanism is a semantic network. The nodes of the network are the basic objects with associated feature values and the links indicate the relations between objects.

The basic objects used in the image description are regions or linear features extracted by automatic segmentation procedures [12, 13]. These procedures produce a set of objects composed of connected regions which are homogeneous with respect to some feature in the input image and long narrow objects which differ from the background on both sides and can be represented as sequence of straight line segments. Only the important objects are described in the model. The automatic image segmentation produces many objects which are not included in the model (as many as 100-300 elements). The model description determines the outcome of the matching procedure and can also be used to guide the segmentation procedure [14].

The description is completed by extracting features of the regions and linear objects. The features are those which can be easily computed from the data and which are reasonably consistent. These features include average values of the image parameters (intensity, colors, etc.), size, location, texture, and simple shape measures (length to width ration, fraction of minimum bounding rectangle filled by the object, perimeter ^{2}/area, etc.). Relations included in the description are those which are easily computed; such as adjacency, relative position, (north of, east of, etc.), near by, and an explicit indication of not near by.

The basic goal for the matching procedure is to determine which elements in the image correspond to the given objects in the model. Most of the objects cannot be recognized based on features alone. They require contextual information to be accurately located. An important idea used by the matching system is to locate a small set of corresponding objects using feature values and weak contextual information. These initial islands of confidence provide the context needed for finding correspondences for the less well defined objects. Finally, when most objects are assigned, the matching can be done solely on the basis of context, i.e., radical changes in a few objects do not cause the matching program to fail.

The basic operation of the matching system is outlined in Fig. 1. In the large outer loop a set of possible matching regions is located for every element in the model. Each of these possible assignments has a rating (probability) based on how well the model and image elements correspond. These ratings are refined by the relaxation procedure in the inside loop, until one or more model elements have one highly likely assignment (usually a probability greater than 0.7 or 0.8). At this point a firm assignment is made and the likely assignments are recomputed using these assigned elements to give the context for the match. The inner relaxation procedure updates the probabilities of the assignment based on how compatible the assignment is with the assignments of its neighbors in the graph (i.e., objects linked by relations). We use a variety of relaxation schemes [1, 8, 9, 10, 11, 15] in this loop, with the criteria optimizing method in [1, 8] giving the best results.

Matching Details

The quality of match between two elements (one each from the model and image or from two different images) is given by the weighted sum of the magnitude of the feature value differences,

$$R(u,n) = \sum_{t=1}^{m} |V_{ut}-V_{nt}| W_t S_t \qquad (1)$$

where u is an element from the model n from the image, m is the number of features being considered, and $V_{ut}(V_{nt})$ is the value of the t^{th} feature of element u(n). W_t is a normalization weight (the same for all tasks) to equalize the impact from all features. S_t is the task dependent strength of a given feature. These strength values distinguish between important, average, and unimportant features. The ratio of the strength values is 5:1 and there is a fourth strength zero which indicates a feature is not used. This rating function is converted to the range [0, 1] by

$$f(u,n) = \frac{a}{R(u,n)+a} \qquad (2)$$

where a is a constant which controls how steep the differences function is. A value of 1 (a sharply declining function) produces the best results with the optimization updating approach. Relations are easily included in this scheme. V_{ut} is the number of relations of type t which are specified in the model and V_{nt} is the number which actually occur in the image. Figure 4 illustrates how these values are computed for a given u_i. For each possible corresponding region n_k, check all u_j (in the model) which are related to u_i to see if the given correspondence (n_ℓ) for u_j is properly related to n_k. When computing the initial probabilities of a match, only those u_j which have been previously assigned can be considered. The basic compatibility measure is computed when given two potential assignments, therefore rather than using all the other units in the model use only the specified unit u_j.

The relaxation procedures require a function which measures the compatibility of a particular assignment n_k with the current assignments at all neighboring (related) units. This is defined by

$$Q_i(n_k) = \sum_{u_j \text{ in } N_i} Q_{ij}(n_k) + \alpha |N_i| f(u_i,n_k) p_i(n_k) \qquad (3)$$

and

$$Q_{ij}(n_k) = \sum_{n_\ell \text{ in } W_j} c(u_i,n_k,u_j,n_\ell) p_j(n_\ell) \qquad (4)$$

Where N_i is the set of objects related to u_i, $|N_i|$ is the number of neighbors, α is a factor between 0 and 1 that adjusts the relative importance of features versus relations (0.1 to 0.25 is the usual range), $p_i(n_k)$ is the current probability for assigning u_i to n_k, W_j is the set of likely assignments of u_j (for efficiency and improved results we generally use only the one most likely assignment here). $c(u_i, n_k, u_j, n_\ell)$ is the same as $f(u_i, n_k)$ except that only relations between u_i and u_j are considered. The vector $\vec{Q}_i$ is normalized to give a probability vector $\vec{q}_i$ which is used by the updating step. The iterative updating is given by

$$\vec{p}_i^{\,(n+1)} = \vec{p}_i^{\,(n)} + \rho_n P_i\{\vec{q}_i^{\,(n)}\} \quad n=0,1,2,\ldots \qquad (5)$$

where ρ is a positive step size to control the convergence speed, P_i is a linear projection operator to maintain the constraint on $\vec{p}_i^{(n+1)}$ that is is a probability vector, and $\vec{q}^{(n)}$ is an explicit gradient function determined by the criteria to be optimized.

$$g_i(n_k) = -q_i(n_k)-p_i(n_k)|N_i|\alpha f(u_i,n_k)(1-q_i(n_k))/D_i$$

$$- \sum_{\substack{u_j \text{ such that} \\ u_i \text{ in } N_j}} \frac{1}{D_j} \sum_{n_\ell \text{ in } W_j} c(u_j,n_\ell,u_i,n_k)(p_j(n_\ell)-p_j\cdot q_j) \qquad (6)$$

where

$$D_i = \sum_{k=1}^{m} Q_i(n_k) \qquad (7)$$

where m is the number of possible assignments.

IV. Extensions to the Matching

This basic matching procedure is able to adequately perform the match for many tasks, but there are extensions which are required for others. These include extensions to handle multiple levels of descriptions for the scene and those to facilitate the image to image matching process.

A. Groups

The matching procedure, as described so far, handles relations between two specific elements, if relations among three or more objects are desired they are specified by combinations of binary relations. They may not yield unique matches, but explicit higher order relations are too expensive to compute and use. We extend the matching and description system to include relations between groups of elements. These groups are specified in the model and can be composed of an object or a collection of separate objects that can be more easily related to others as a group. For example, in Fig. 2, the area of San Francisco can be considered as a group composed of the urbanized area, and the park-like areas. The bay, bridges, and islands can form another group. The separate clusters of storage tanks or buildings could be used to form groups in Fig. 3.

We make several assumptions about the groups of objects. (1) The components of a group are spatially close, not widely scattered through the image. (2) Relations (adjacency, above, etc.) between elements within a group are meaningful, but usually not between individual elements in two separate groups. (3) Relations between groups are consistent and predictable. (4) The feature values for individual objects relative to the averages for the group are well defined (e.g., intensity greater than average, x location in the top fifth). This easily handles structures in aerial images and might be extended to three-dimensional structures possibly with some changes in assumptions.

Thus we simply extend the basic network description of the model to include for each element a pointer to the group, and feature values relative to the group average values, with relations between the groups specified as links between the group nodes. Group features are not available for the image description until the correspondences are located. Initial groupings could be computed in limited cases by creating sets of objects where each is near at least one other member of the set. We could consider groups as descriptions at higher levels in a generalized pyramid structure [16], but our description of the higher level object is based solely on the lower level descriptions of its parts rather than the description of the object at lower resolutions. For a description which encompasses more than two levels, a general multilevel description should be used, but a matching scheme would require a means for linkage between levels.

These group features and relations are incorporated into the matching procedure much the same as the initial features and relations (Eqs. 1-6). But, we apply a second relaxation step in the inner loop (see Fig. 1) using only the group features and relations to compute the compatibility measures. The average feature values and the location of each group are computed from the current most likely assignment for each of the components of the group (i.e., the top one after the previous relation updating step). Figures 4 and 5 illustrate how group relations enter in compatibilities. As illustrated in Fig. 4 the measure for standard relation is given by whether the relation specified in the model between two elements actually occurs between the two possible assignments. The test for group relations is a bit different. The compatibility measure $(c(u_i, n_k, u_j, n_\ell))$ can be computed only for u_i in group G_i and u_j in group G_j where G_i is related (is a neighbor in the graph) to G_j. The problem is then to determine if n_k is properly related to G_j (e.g., above) and G_i is also related to n_ℓ (above). $R(u, n)$ as given in Eq. 1 is computed in the same manner except all possible second model units (u_j) are considered. (Possible in this case means that the two groups are related and u_j is assigned.)

The relations between groups are specified by the model and the test between n_k and G_j must be computed each time since specifications of the group (location, extent, etc.) may change on every iteration. The relations between simple elements in the model should correspond to relations between elements in the segmentation of the image so that these can be computed once and stored. This difference results in an increased computation time for relations between groups compared to relations between basic elements.

The group information can be used in othre important ways. The matching procedure operates on every element in the model description at every step, but once a group has been completed there is no need to repeatedly consider the elements in the group for additional assignments. It is not the case that once an individual element is assigned it can be removed for consideration since succeeding iterations are needed for correcting any errors which occur. The group elements are not eliminated from the entire process - the relations to other elements are even more important than before - they are simply given fixed probability vectors (ones for all current firm assignments). Thus the main computation steps (Eq. 1-7) are not needed for these elements which reduces the computation time proportionally. By adding this assigned group elimination to the matching procedure the time is reduced by about half and the results remain the same.

B. Image to Image Matching

Matching of images at a symbolic level can provide information similar, though not identical, to pixel level image matching. The result is a set of pairs of corresponding objects. From these it is easy to extract global transformations (scale, position, orientation, intensity shifts, etc.), relative displacements (for relative object heights), and local object changes. The matching system is identical to that used for the model to image matching, but there are some differences in how it is used.

Some of the differences are caused by the differences in the nature of the descriptions of images and scene models. The scene model contains only important objects and only those feature values and relations which are relevant or consistent. The image segmentation cannot be restricted in the same way, thus there are many extra unimportant regions, all feature values are available for all regions and all possible relations between two regions are included in the description.

The increased number of regions is addressed first, rather than trying to find a match for all regions in an image we can restrict the search to those regions which meet a given criterion. We can filter the image to eliminate ill formed regions (using the shape parameters with very loose thresholds), or can restrict the match to some other subset of regions (the brightest half).

The availability of all features is more a benefit than a liability. We can use absolute locations as very strong features, after initial matches are located which can provide the necessary transformations (translation, etc.). By using absolute position, the matching can be performed when differences occur in image segmentations and feature descriptions. Initially the absolute position cannot be used in the matching since we allow arbitrary translations, but when several matches are located we can generate global transformations which will approximately register the images. Because of distortions, height differences, segmentation differences, etc., no global transformation will work perfectly, but the object positions can be used as important features. This is implemented by adding a transformation generation step prior to the determination of initial likelihoods. The transformation is generated using the objects with translations closest to the mean translation. This selection can be done in many ways with different degrees of complexity, we chose a simple method since we do not require subpixel level accuracy in the location transformation. The strengths of the position features are increased from low to medium to high as more correspondences are located. Additionally the number of iterations to try before termination must be reduced when there are few (less than 10) regions remaining to be assigned.

The final change for image to image matching is to perform the match in both directions independently. This means that when we match two images A and B we treat A as the model and find the correspondences in B, then treat B as the model and find the correspondence in A. The final result is all the pairs of regions which are located in both cases. This eliminates a few correct matches which are found in one case but also eliminates most (all in the examples) incorrect matches since these are predominantly caused by segmentation differences (combined or missing regions).

C. Local Matching

The matching procedure usually locates clear matches first and then builds on these islands of confidence. The building steps generally require few (i.e., one) iterations. This fact can be used to substantially improve the computation time of the matching procedure. If we consider only those elements which are strongly related to already assigned objects and allow very few (e.g., one) iterations, it is possible to quickly and cheaply make many assignments. We limit the number of iterations on this step to one since we are using only very local information which is much less reliable than the full global description. This means that we are very confident in the assignment which arises - there are no competing possibilities. More assignments are made if more iterations are allowed, but many more errors are introduced. When thre are no unassigned objects strongly related to ones that are assigned or no quick assignment is found then the procedure reverts to the normal matching. This can also be applied with the group elimination operation reducing the time even more. When using the two step hierarchical matching process we only use the regular features and ship the group step isnce we are concentrating on strong local information only. The determination of strongly related elements can be arbitrarily complex, but we have chosen to limit it to a simple test of adjacency or nearby, since many of the relative position relations are already limited to these kinds of regions (especially in the model description).

V. Results

We have applied this system to a variety of images (generally two views of each scene, see Fig. 2, 3). For different views of the same scene, we use the same model. The results are presented as overlays on the original images, showing the border of regions or center lines of linear features. The labels are taken from the name given in the model, either the user derived model or the image which serves as a model. Table 1 summarizes the results.

Figure 6 shows the results of matching the model to two images of San Francisco area (Fig. 2). The errors in the second view (Fig. 6b) are caused by the segmentation errors. The two sections of the Bay Bridge are missed by the linear features program and this causes these two to be missed plus the island which is adjacent to the bridges and both portions of the bay is mismatched. (Note that the two sections of the bay were intended to be split by the bridges.) Figure 7 is the same result except that the group features and relations are used. The results are the same except that one section of the bay bridge in view 2 is not matched (which is the correct result) and a second match is found for a park area in view 1. The computation times (with and without group features) are similar.

Figure 8 gives the results for a subwindow of the low altitude aerial images (Fig. 3) without the group information. Figure 9 shows the improvement when group features and relations are used. In the first view 2 fewer mistakes are made. In the second view mistakes are reduced by 7 and correct matches are increased by 3. Because of the cost of group relations the computtion time increased substantially. Different objects are segmented poorly in the two views, but the matching still works well for both. In the seven errors (see Table 1), 3 are objects with no correct match, 3 are multiple matches where the correct match also occurs and one is an extra match to a small nearby region. Figure 10-12 illustrate the image to image matching process. In Fig. 10 the first view A is used as the model, and the second is used in Fig. 11 (the image used as the model is the one on the left). Figure 12 shows those pairs which occur in both cases. Table 2 gives the computed disparities for each of these 31 matched objects.

VI. Summary and Conclusions

This paper presents an extension to an earlier symbolic matching program. The extensions improve the performance of the matching procedure for model to image matching when there are groups or clusters of objects. Additional changes improve the performance of the image to image matching task. The matching results are very good, but not perfect. There is no post processing to eliminate matches which are not consistent with the others which could reduce the errors.

VII. References

[1] O. Faugeras and K. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," IEEE T-PAMI, Vol. 3, No. 6, Nov. 1981, pp. 638-642.

[2] H. Moravec, "Rover Visual Obstacle Avoidance," Proc. 7-IJCAI, Vancouver, B.C., Canada, Aug. 1981, pp. 785-790.

[3] H. Baker and T. Binford, "Depth from Edge and Intensity Based Stereo," Proc. 7-IJCAI, Vancouver, B.C., Canada, Aug. 1981, pp. 631-636.

[4] S. Barnard and W. Thompson, "Disparity Analysis of Images," IEEE T-PAMI, Vol. 2. No. 4, July 1980, pp. 333-340.

[5] C. Wong, H. Sun, S. Yada, and A. Rosenfeld, "Some Experiments in Relaxation Image Matching Using Corner Features," Univ. of Maryland, Computer Vision lab, Computer Science Center, TR-1071, 1981.

[6] C. Clark, D. Conti, W. Eckhardt, T. McCulloh, R. Nevatia, and D. Tseng, "Matching of Natural Terrain Scenes," in 5-ICPR, Miami, Fla., Dec. 1980, pp. 217-222.

[7] D. Gennery, "A Feature Based Scene Matcher," in 7-IJCAI, Vancouver, B.C., Canada, Aug. 1981, pp. 667-673.

[8] O. Faugeras and M. Berthod, "Improving Consistency and Reducing Ambiguity in Stochastic Labeling: An Optimization Approach," IEEE T-PAMI, Vol. 3, July 1981, pp. 412-424.

[9] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene Labeling by Relaxation Operations," IEEE T-SMC, Vol. 6, June 1976, pp. 420-453.

[10] S. Peleg, "A New Probabilistic Relaxation Scheme," IEEE T-PAMI, Vol. 2, No. 4, July 1980, pp. 362-369.

[11] R. Hummel and S. Zucker, "On the Foundations of Relaxation Labeling Processes," in 5-ICPR, Miami, Fla., Dec. 1980, pp. 50-53.

[12] R. Ohlander, K. Price, and R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Comp. Graphics and Image Proc., Vol. 8, pp. 313-333, 1978.

[13] R. Nevatia and K. Babu, "Linear Feature Extraction and Description," Comp. Graphics and Image Proc., Vol. 13, 1980, pp. 257-269.

[14] K. Price, and G. Medioni, "Segmentation Using Scene Models," to be published.

[15] L. Kitchen, "Relaxation Applied to Matching Quantitative Relational Structures," IEEE T-SMC, Vol. 10, Feb. 1980, pp. 96-101.

[16] A. Hanson and E. Riseman, "VISIONS: a Computer System for Interpreting Scenes," in Computer Vision Systems, A. Hanson and E. Riseman (eds.), Academic Press, New York, 1978, pp. 303-333.

TABLE 1. SUMMARY OF MATCHING RESULTS

Figure	"Model"	"Image"	Correct	Wrong	Macro Iterations	Time
8 left	Model	View 1	35	5	27	1:25
8 right	"	View 2	32	11	23	1:18
9 left	"	View 1	35	3	16	1:03 Group
9 right	"	View 2	35	3	17	1:11 Group
10	View 1	View 2	34	5	41	1:21
11	View 2	View 1	36	4	50	1:42
12	Combine 1,2		33	-	-	-
6a	Model	View 1	15	0	8	:29
6b	"	View 2	12	3	8	:36
7a	"	View 1	16	0	8	:34 Group
7b	"	View 2	12	2	8	:37 Group

TABLE 2. TRANSLATIONS COMPUTED FROM MATCHING REGIONS. THESE ARE GROUPED BY THE CLUSTERS (TOP TO BOTTOM) TO SHOW SIMILARITIES AMONG NEARBY OBJECTS

Region ID	ΔR	ΔC	Which cluster	
8	91.0	195.7	1	
21	89.4	197.6	1	
22	90.0	198.2	1	
32	89.6	196.4	1	
31	89.7	196.0	1	
10	90.4	197.8	1	
18	89.9	198.8	1	
26	88.7	195.9	1	
23	89.6	197.4	1	
9	87.5	198.1	1	
17	89.5	194.2	1	
Group range	3.5	4.6		
35	84.2	193.7	2	
30	85.7	198.6	2	
3	77.0	187.7	2	Region broken in half
12	83.7	198.7	2	
Group range	2	5		
36	79.7	200.6	3	
11	80.5	201.3	3	
39	81.9	200.7	3	
25	80.7	201.0	3	
7	79.7	199.9	3	
19	79.6	202.3	3	
20	81.4	199.4	3	
13	78.2	202.8	3	
14	77.2	200.0	3	
4	77,6	202.1	3	
Group range	4.7	3.4		
29	77.6	193.6	4	
15	77.2	193.8	4	
16	79.1	198.2	4	
34	76.4	194.1	4	
2	77.4	194.0	4	
5	76.5	193.8	4	
Group range	2.7	4.6		
Overall range	14.6	9.2		

TABLE 3. COMPARISON WITH OTHER RELAXATION TECHNIQUES. COMPLETE IS THE OPTIMIZATION APPROACH [1, 8]. PROJECT ONLY USES THE PROJECTION FUNCTION BUT NOT OPTIMIZATION. PRODUCT COMBINES THE INDIVIDUAL MATCHING VALUES USING PRODUCE RATHER THEN SUM (AS IN EQ. 4). ORIGINAL IS FROM [9] AND IS GIVEN FOR HISTORICAL PURPOSES.

Scene	Relaxation Type	Right	Wrong	Iterations	Time	
Top group	Complete	14	0	10	:07	
of	Project only	14	0	11	:07	
14	Product combination	14	3	4	:02	
	Original updating	14	3	13	:09	
High altitude	Complete	15	0	8	:29	
model	Project only	7	0	5	:21	
with	Product	18	4	11	:27	(14 on iteration 1)
view 1	Original	14	8	15	time limit	
Low altitude	Complete	35	3	17	1:11	
model	Project only	32	2	11	1:20	
with	Product	33	12	4	:09	(30-6 on iteration 1)
view 2	Original	38	8	11	1:45	
Low altitude	Complete	34	5	41	1:21	
view 1	Project	24	3	21	1:13	
with	Product	35	43	20	0:40	Very quick decisions
view 2						
View 2	Complete	36	4	50	1:42	
with	Project	23	2	14	0:55	
view 1	Product	36	33	12	0:23	

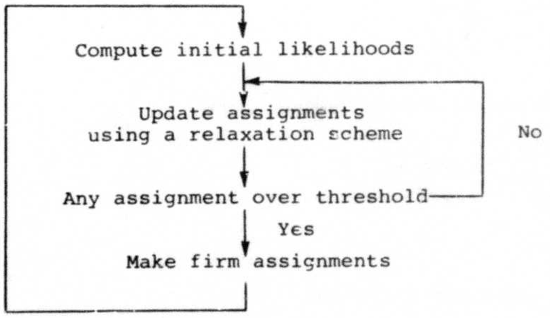

Compute initial likelihoods
Update assignments using a relaxation scheme
No
Any assignment over threshold
Yes
Make firm assignments

Figure 1. Overview of symbolic matching system.

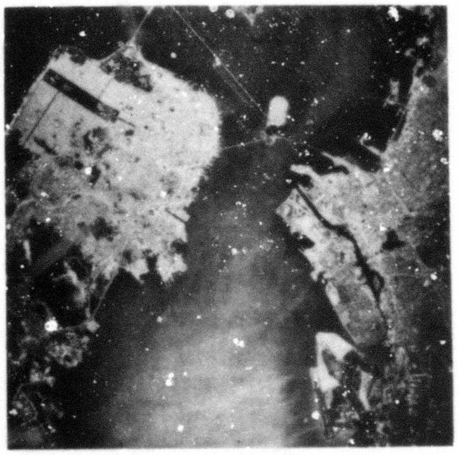
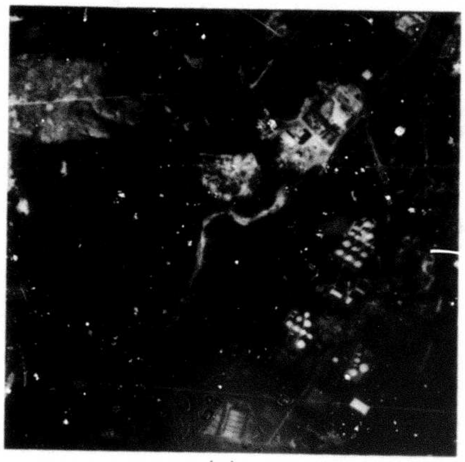


Figure 2. High altitude view of San Francisco area.


(a)

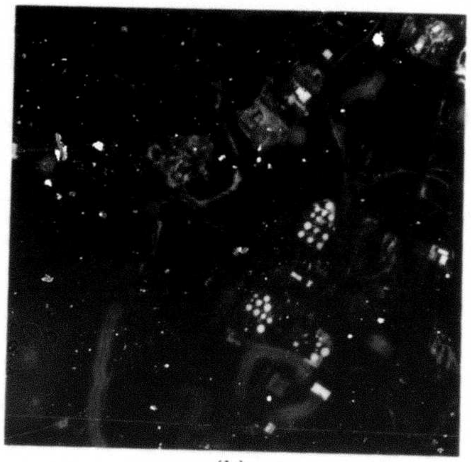

(b)

Figure 3. Low altitude aerial image: (a) view 1 (October), (b) view 2 (August).

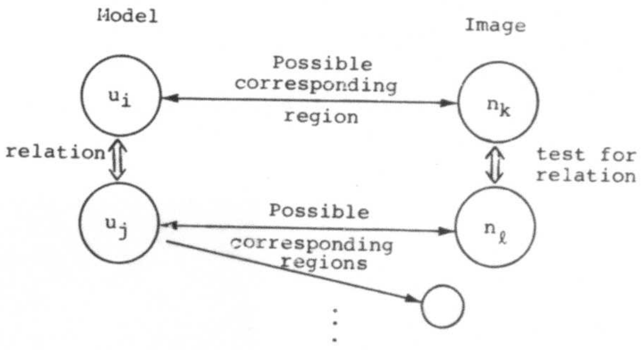


Figure 4. Use of relations in compatibility computation.

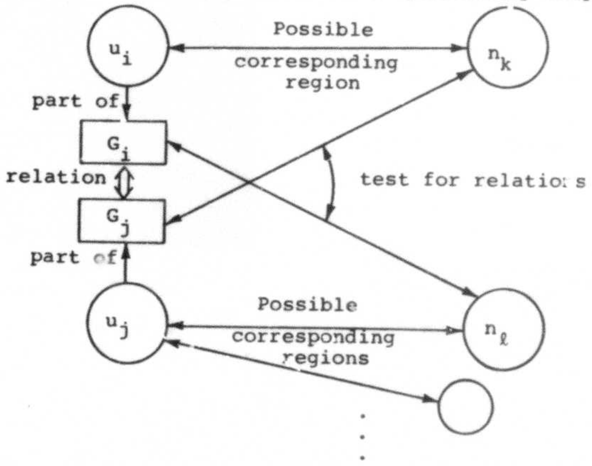
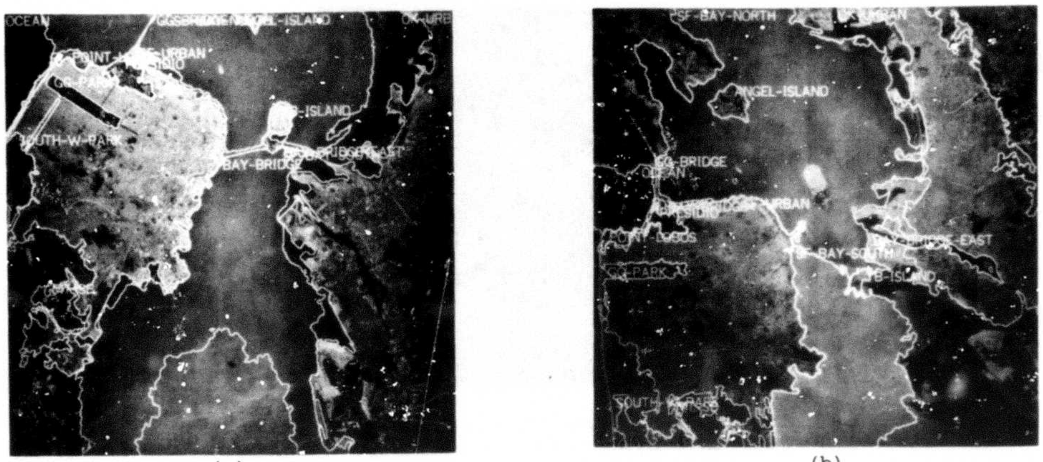
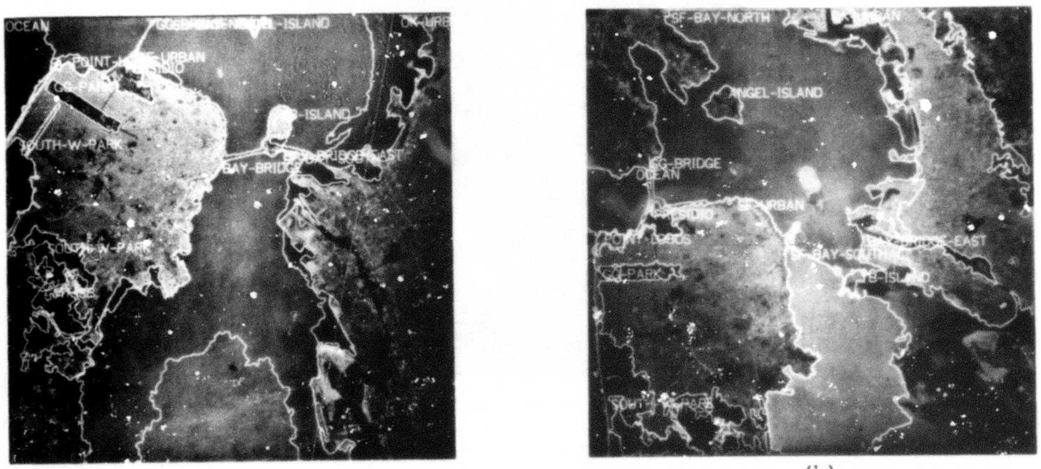


Figure 5. Use of relations between groups in compatibility computation.



(a) (b)

Figure 6. Matching results without group features: (a) view 1, (b) view 2.



(a) (b)

Figure 7. Matching results group featues: (a) view 1, (b) view 2.

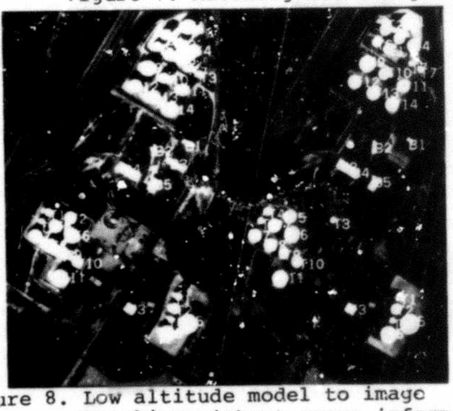


Figure 8. Low altitude model to image matching without group information. Left: view 1, right: view 2

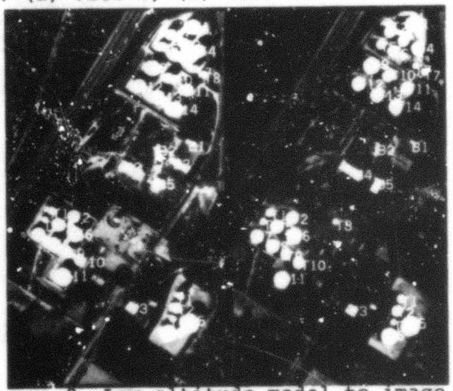


Figure 9. Low altitude model to image matching using group features. Left: view 1, Right: view 2

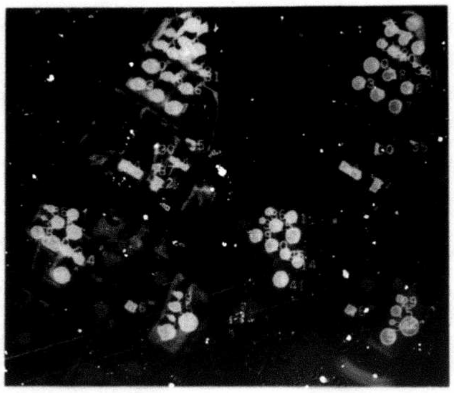


Figure 10. Low altitude image to image (view 1 used as model) matching. Left: view 1, Right: view 2

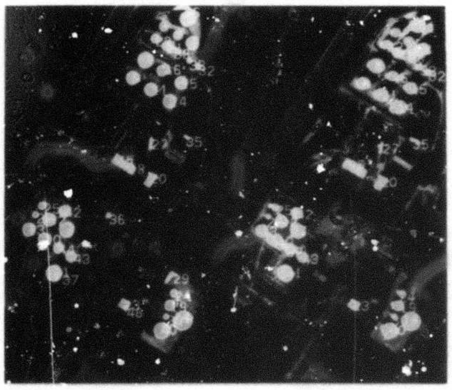


Figure 11: Image to image (view 2 used as model). Left: view 2, Right: view 1

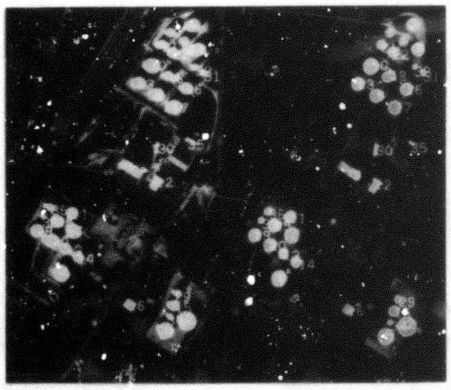


Figure 12. Image to image matching, the combined results of Fig. 10 and Fig. 11. Left: view 1, Right: view 2

AD-P000 138

Automatic Generation of Depth Maps from Stereo Images

Bruce D. Lucas
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

This paper describes an efficient technique for calculating depth maps from stereo pairs of images. This technique is based on a linear approximation to the image at each point, and was described in another form in (Lucas 81). We show how the algorithm can be augmented by smoothing the images, iterating the calculation, and using weighting factors. A theoretical result concerning the performance of the algorithm is presented, and experimental results on random dot stereograms and natural images are described.

1. Introduction

In (Lucas 81), we presented an efficient technique for stereo matching based on the use of derivatives to direct the search for the best match. As presented, the technique was able to find matches for isolated points, for example feature points located by an interest operator (see (Moravec 80)). For some applications it is desirable to know the distance at a dense set of points or at each pixel in the image. For example, image segmentations based on depth (distance) information as well as light intensity would not cause spurious regions due to surface markings to be generated. Another application is the generation of topographic maps from aerial stereo images. The importance of the depth map as an intrinsic image in image interpretation was recognized in (Barrow 78). In this paper we show how the technique presented in (Lucas 81) can be extended for the direct computation of a depth map from a stereo pair of images.

A depth map can be defined as an "image" in which the pixel at each point encodes the distance of the image at that point from the camera. What we shall actually compute is the closely related disparity map. The value $h(x,y)$ at position (x,y) in a disparity map is a statement that the image at position (x,y) in (say) the left image of a stereo pair corresponds to the image position $(x + h(x,y), y)$ in the right image. (What would actually be stored in the depth map would be a linear function of $h(x,y)$ that maps the relevant range of disparities onto the available range of pixel values.) That is, $h(x,y)$ encodes the disparity between the left and right images at position (x,y). The computation of the disparity, or equivalently the image correspondence, is the central problem of stereo image interpretation. Given the disparity and the relative camera positions, calculating the distance is a simple matter of geometry. It should be noted that the definition of disparity given above assumes that certain of the camera axes are parallel. If this assumption is violated, the particulars of what follow change, but the algorithm itself is not materially changed. For example, it is always possible to resample the images producing a new pair of images which do satisfy the assumption.

2. Calculating the depth map efficiently

Let $L(x,y)$ be the pixel value at position (x,y) of the left image of a stereo pair, and $R(x,y)$ be the pixel value at position (x,y) of the right image. As implied by our definition above, assume that $L(x,y) = R(x + h(x,y), y)$. Our goal will be to compute an estimate of the depth map, $\hat{h}(x,y)$, such that $L(x,y)$ is as nearly equal as possible to $R(x + \hat{h}(x,y), y)$. Now, one way to do this would be just to try all possible values of $\hat{h}(x,y)$ at each (x,y), finding the one which minimizes $|L(x,y)-R(x+\hat{h}(x,y),y)|$. This approach has two problems. First, it is inefficient. But more importantly, it is now possible for $\hat{h}(x,y)$ to vary wildly as x and y vary. But we know that $h(x,y)$ will vary smoothly at most points, due to the coherence of matter. (See for example (Marr 79)). Thus we need to constrain $\hat{h}(x,y)$ to vary relatively smoothly. We will do this implicitly by assuming in our solution that $h(x,y)$ is nearly constant in a small neighborhood about each (x,y). In particular, we will choose each $\hat{h}(x,y)$ to minimize the local error

$$E(x,y) = \sum_{u,v\ near\ x,y} [R(u+\hat{h}(x,y),v)-L(u,v)]^2. \quad (1)$$

(All subsequent sums will be over the same range.) This relies on assuming that $h(x,y)$ will be nearly uniform over the region of summation around (x,y). Or, looked at in another way, our computation of $\hat{h}(x,y)$ will incorporate information from a region around (x,y). Neighboring values of $\hat{h}(x,y)$ will incorporate information from nearly identical neighborhoods of the image, and thus can be expected to vary but a small amount. This will be seen in more detail below.

We now turn to the problem of calculating $\hat{h}(x,y)$ efficiently. Here is where the technique described in (Lucas 81) comes in. To minimize $E(x,y)$ in equation (1), we first approximate $R(u+\hat{h}(x,y),v)$ as $R(u,v)+\hat{h}(x,y)R_x(u,v)$, where R_x is the derivative of R with respect to x, obtaining

$$E(x,y) \cong \sum [R(u,v)+\hat{h}(x,y)R_x(u,v)-L(u,v)]^2. \quad (2)$$

But this is just a sum of terms quadratic in $\hat{h}(x,y)$. Thus, we can differentiate $E(x,y)$ with respect to $\hat{h}(x,y)$ and set the result equal to zero, obtaining

$$0 = \sum [R(u,v)+\hat{h}(x,y)R_x(u,v)-L(u,v)]R_x(u,v). \quad (3)$$

Solving,

$$\hat{h}(x,y) = \frac{\sum [L(u,v)-R(u,v)]\ R_x(u,v)}{\sum R_x(u,v)^2}. \quad (4)$$

Now we see in particular how $\hat{h}(x,y)$ is constrained to be smooth. Each of the sums in the numerator and in the denominator of equation (4) is roughly a smoothed version of an "image" similar to the original images $R(x,y)$ and $L(x,y)$. The ratio should be similarly smooth except where the denominator is near zero. An inspection of the denominator shows that this occurs only where the image is relatively constant; in such cases the derivative of the image gives no information and so the method is not expected to work anyway. Use of this algorithm in a vision system should take this point into consideration.

From the formula above, we can see that $\hat{h}(x,y)$ can be calculated efficiently if we take "(u,v) near (x,y)" to mean "(u,v) in a rectangular neighborhood around (x,y)". The efficiency stems from a well-known incremental technique for smoothing an image, i.e. calculating $\Sigma f(u,v)$ where (u,v) ranges over a rectangular region around a point (x,y), using only two additions and two subtractions per pixel. (See, for example, (Price 76)). Thus, $\hat{h}(x,y)$ above can be calculated using only two additions, four subtractions (including one to compute $R_x(x,y)$), two multiplications, and one division per pixel.

3. Improving the calculation

This technique does not perform entirely satisfactorily as given. In particular, it cannot tolerate a very large disparity. However, as discussed in (Lucas 81), three things can be done to make it a viable algorithm.

First, the accuracy of the linear approximation made in equation (2) can be improved by smoothing $L(x,y)$ and $R(x,y)$. This can also be thought of as removing high spatial frequency components from the images. The value of this is shown by the following observation: If we are attempting to find the disparity between, say, two identical sine waves, then a disparity of one-half the wavelength of the sine waves results in an inherently ambiguous match, and for any larger disparity, the closest match is incorrect. Thus, low spatial frequencies can be matched over a wider disparity range, so removing high spatial frequencies should increase the range of unambiguous match. Consider, for example, the problem of matching two buildings textured with windows. If the disparity is greater than the size of the windows, then a mismatch in which two different windows are paired is likely. However, if we first smooth the images, blurring out the windows and leaving only the buildings, then we should expect to be able to match the buildings, given disparities up to about half the distance between buildings, at which point we may begin to incorrectly pair the buildings. If such large disparities need to be tolerated, we can blur the images even more and match groups of buildings, and so on. This reasoning is made more precise in the next section.

The second improvement is based on two observations: first, $\hat{h}(x,y)$ as computed above is only an approximation to $h(x,y)$, and second, the blurring of the images has blurred out detail which could provide a more accurate match. For example, if we blur together two details whose disparity (depth) is different, the best we could hope to calculate is a disparity somewhere midway between the two. Taken together, these observations suggest that we should iterate the disparity calculation, using the previously calculated $\hat{h}(x,y)$ as an initial guess at each stage, and that we should use less blurred versions of $L(x,y)$ and $R(x,y)$ at each iteration. The calculation now becomes

$$\hat{h}_{k+1}(x,y) \equiv$$

$$\frac{\sum [L(u,v)-R(u+\hat{h}_k(x,y),v)]\, R_x(u+\hat{h}_k(x,y),v)}{\sum R_x(u+\hat{h}_k(x,y),\, v)^2}, \quad (5)$$

where $L(x,y)$, $R(x,y)$ and $R_x(x,y)$ are actually blurred versions of the left and right images.

The third improvement is also discussed in (Lucas 81). It is based on the observation that the linear approximation used in equation (2) is more or less accurate at various places in the pictures, and that the accuracy can to some extent be detected. The accuracy can be accounted for by weighting the contribution of each point (x,y) in the sums in equations (4) and (5) by an estimate of the accuracy of the linear approximation at that point. The effect of this weighting is not to extend the range of acceptable disparities, but rather to improve the accuracy of the calculated estimate of the disparity. For more details see (Lucas 81).

4. An analytical result

An analytical result concerning the effect of filtering the images is possible. Consider a one-dimensional version of the problem: given $L(x)$ and $R(x) = L(x+h)$, calculate h. The estimate $\hat{h}$ analogous to the one in equation (4) is

$$\hat{h} = \frac{\sum [L(u)-R(u)]\, R_x(u)}{\sum R_x(u)^2}. \quad (6)$$

Now, by the Fourier theorem, $L(x)$ can be represented by

$$L(x) = \sum_k r_k \cos 2\pi kx + \theta_k, \quad (7)$$

where the r_k constitute the (square root of) the power spectrum of $L(x)$ and the θ_k constitute the phase spectrum of $L(x)$. It can be shown that the estimate for h given in equation (6) is equivalent to

$$\hat{h} = \frac{\sum_k k r_k^2 \sin 2\pi kh}{2\pi \sum_k k^2 r_k^2}. \quad (8)$$

Thus, the estimate $\hat{h}$ depends only on the actual h and the power spectrum of $L(x)$. This is a highly interesting result, because the power spectra of typical images tend to be similar, *while all the information is contained in the phase spectrum* (see (Oppenheim 81)). Thus it should be possible to make general statements about the behavior of the algorithm on classes of images. Examination of this formula bears out the suspicion that suppressing high spatial frequencies improves the estimate of the disparity.

5. Experimental results

Depth maps were obtained from successive iterations of the algorithm applied to a random dot stereogram with a raised square in the middle. The stereogram is shown in figure 1. The resulting depth maps are shown in the left halves of figures 2 through 5. The depth map is shown as an "image" with the gray value encoding the disparity. Each successive figure shows the results of applying the algorithm to successively less blurred versions of the images, using previous depth maps for initial estimates. On the right side of each image is shown a "reliability map" which indicates the reliability of the depth map at that point; white indicates high reliability, black low reliability. The reliability at each point is based on the sum of the weighting factors mentioned in section 3. Note the low reliability computed at the boundary of the square, where the region of summation at each point includes regions of different disparities. Note also the thick bar of unreliability at the left side of the square where the background behind the square is not visible to both eyes and therefore the disparity is undefined.

The success of the algorithm has also been demonstrated on random-dot stereograms with smoothly-changing disparities. Some success has also been attained on natural images, although such results are difficult to assess because of the lack of ground-truth data.

6. Conclusions and Future work

A technique has been demonstrated for efficiently calculating the depth map based on a linear approximation to the image at each point. The algorithm has been further improved by the addition of smoothing, iteration, and weighting. A theoretical result has been presented to show why smoothing helps. Experimental results have been presented which demonstrate that the algorithm works for random dot stereograms. Preliminary evidence shows promise for natural scenes.

The algorithm can be improved in several ways. As stated, it does not do well in the vicinity of depth discontinuities. Moreover an initial estimate must be provided in some way. This suggests combining the algorithm with an algorithm based on feature points to provide the initial estimate. As stated, the algorithm does not deal well with brightness variations between equivalent portions of the left and right images (due to film and camera differences, and to specular reflections). However, a technique was given in (Lucas 81) for dealing with this problem. As stated above, the algorithm may fail in featureless regions, although these can be detected (by a small denominator in equation (4)). What to do at such points is not clear. Finally, the theoretical results presented in section 4 should be extended to the weighted case.

Bibliography

Barrow, H. G. and J. M Tenenbaum. "Recovering Intrinsic Scene Characteristics from Images," *Computer Vision Systems.* (A. Hanson and E. Riseman, eds.). Academic Press 1978. pp. 3-26.

Lucas, Bruce D. and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision." Proceedings of the Seventh International Joint Conference on Artificial Intelligence, August, 1981.

Marr, D. and T. Poggio. "A Computational Theory of Human Stereo Vision." *Proceedings of the Royal Society of London B-204* (1979), 301-328.

Moravec, Hans P. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Tech. Rept. CMU-RI-TR-3, Carnegie-Mellon University Robotics Institute, September, 1980.

Oppenheim, Alan V. and Jae S. Lim. "The Importance of Phase in Signals." *Proceedings of the IEEE 69*, 5 (May 1981), 529-541

Price, K. "Change Analysis and Detection in Multi-Spectral Images," Appendix. Ph.D. Thesis, Carnegie-Mellon University, 1976.

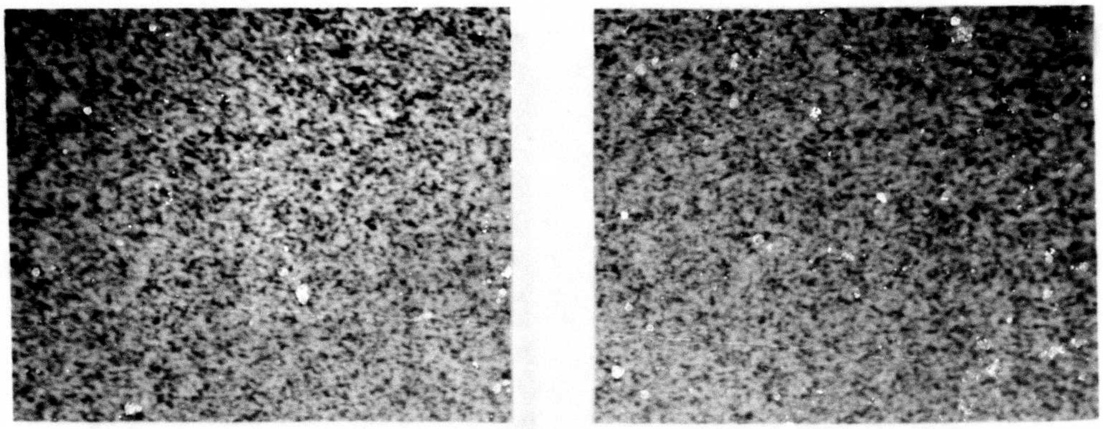


Figure 1.

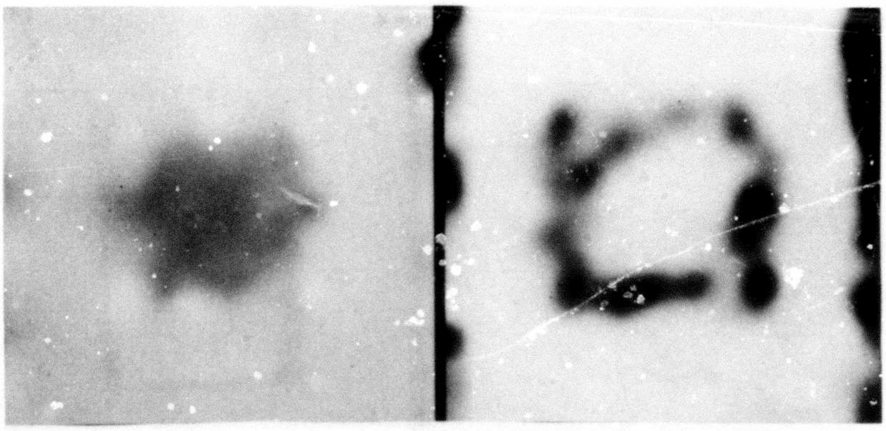


Figure 2.

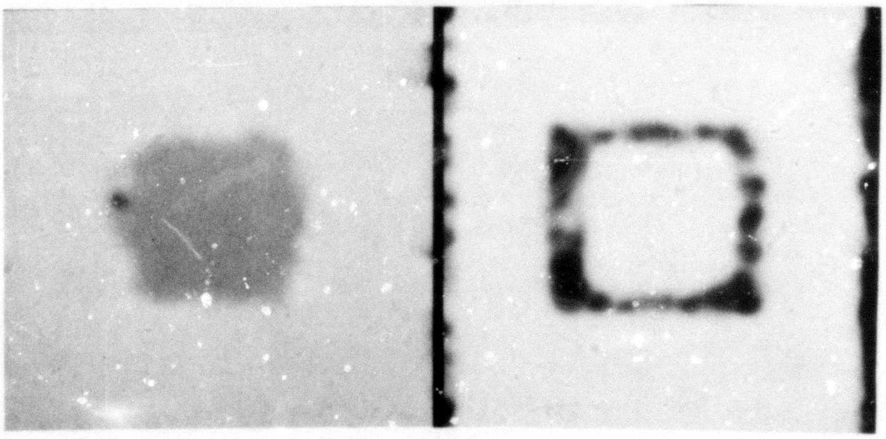


Figure 3.

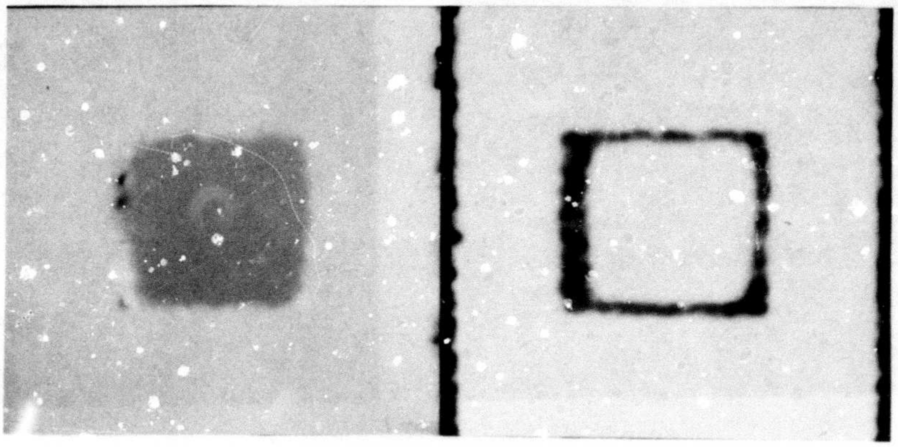


Figure 4.

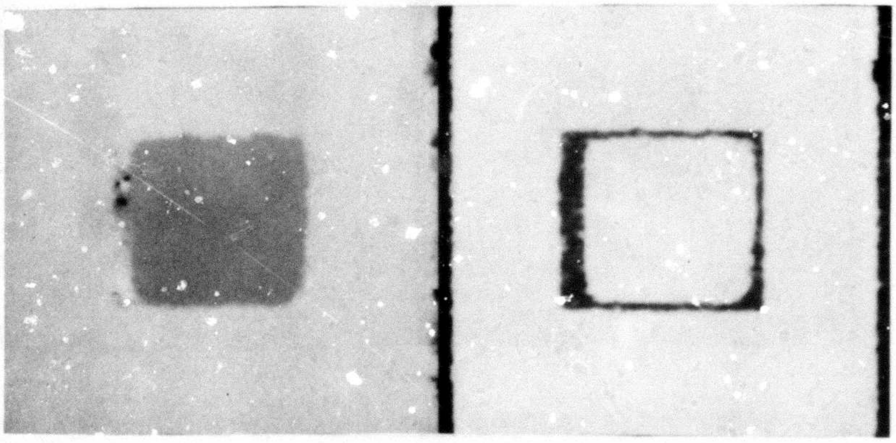


Figure 5.

AD-P000 139

Analysis of Low-level Computer Vision Algorithms for Implementation on a VLSI Processor Array

Michael R. Lowry and Allan Miller

Stanford Artificial Intelligence Laboratory
Stanford University, Stanford, CA 94305 USA

Abstract

In a recent paper [Lowry 81], we described an architecture for a computer vision rectangular processor array that is suitable for VLSI implementation. In this paper we will review that architecture and discuss extensions to it and present results of an array simulator applied to vision algorithms. We will also present an algorithm for re-routing an array with bad processors into a working subset of the array, making it feasible to implement a large array on one wafer-sized chip.

Overview

Several groups have implemented rectangular arrays of parallel processors for image processing applications [Duff 81]. Most of these processors have been designed for implementation using only a few processing elements per chip. For reasons of economy, size, low power consumption, reliability, speed, and simplicity of design, our approach is *to implement an entire array of relatively simple processors on one wafer-size chip.*

Using a single wafer implementation results in several benefits. By keeping signal lines on-chip fan out is decreased thus enhancing speed and density while lowering power consumption. A wafer implementation also makes the best utilization of smaller feature sizes, since higher density makes possible implementation of a larger array and speeds up switching times proportionately. At present the number of processors that can be put on a sub-array is limited to the number of pins on an IC package, unless communication between processing elements on different chips is time multiplexed. Thus there is no intermediate step between small sub-arrays of processing elements and a single wafer-scale implementation of an entire array. A single wafer size chip also provides orders of magnitude savings in power and space, which can be particularily important for autonomous and mobile applications.

Architecture

The architecture we are describing is a synchronous bit-serial SIMD rectangular array with 4-neighbor connectivity. The array size is matched to the image being processed, simplifying control of the array and reducing the amount of memory that each processing element needs. Control is in the form of microcode (register-transfer level) signals shared among all processing elements.

Figure 1 shows the design of one processing element. The processing element is bit-serial, so the data paths in the figure are one bit wide. Memory is implemented as a set of serial shift registers, shifting from MSB to LSB with the MSB coming from a data selector and the LSB going to the arithmetic unit. There are two 8-bit D registers for holding operands and a 16-bit R register for storing a result. The arithmetic unit is a full adder with the carry feeding back into the adder through a register. Operands for the adder come from two input multiplexors that can select from the constants "one" or "zero", the D registers or the R register, or the sum of the arithmetic unit of a neighboring processing element (through a register). Results from the adder go to an output selector where they can be routed to any of the registers. The output selector loads the MSBs of all registers; one register can be loaded from the adder output while the other registers are loaded from their own LSBs. This allows registers to be used as operands without losing their original contents. For doing conditional

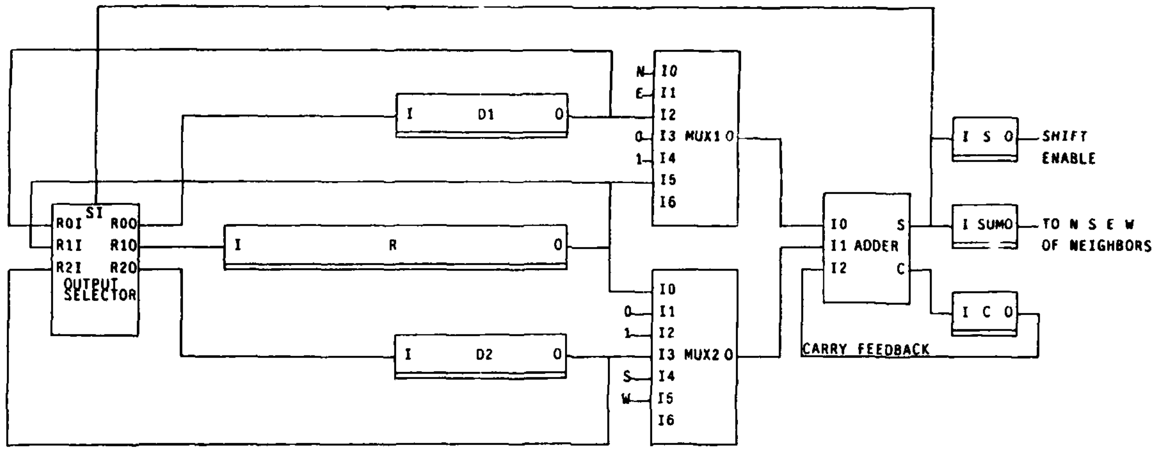


Figure 1. Processing element design.

operations, there is an S register that can be loaded from the adder output. If the S register is a "zero", shifting is disabled in the data registers regardless of the state of the control, effectively disabling the processing element. The processing element is controlled by signals common to all processing elements. Control signals can route adder inputs and outputs, load the S register, and independently shift data registers.

This processing element was implemented in nMOS and fabricated (see figure 2) at the Stanford integrated circuit facilities in late 1980. The chip was tested and found to be operational using an F-15 module test stand at Hughes Aircraft Company. An analysis based on Carver and Mead's design rules [Mead 80] indicates that the chip should execute internal operations at a 10 MHz clock rate. However, limitations inherent in the tester precluded testing the maximum clock rate.

Algorithm Feasibility

An examination of algorithm feasibility was done using the Grinnell image processor to expose problems that would arise in detailed implementation of various vision algorithms in a processor array. Although these experiments are preliminary, the resulting algorithm implementations run much faster and have generated new interest in special implementations of vision algorithms.

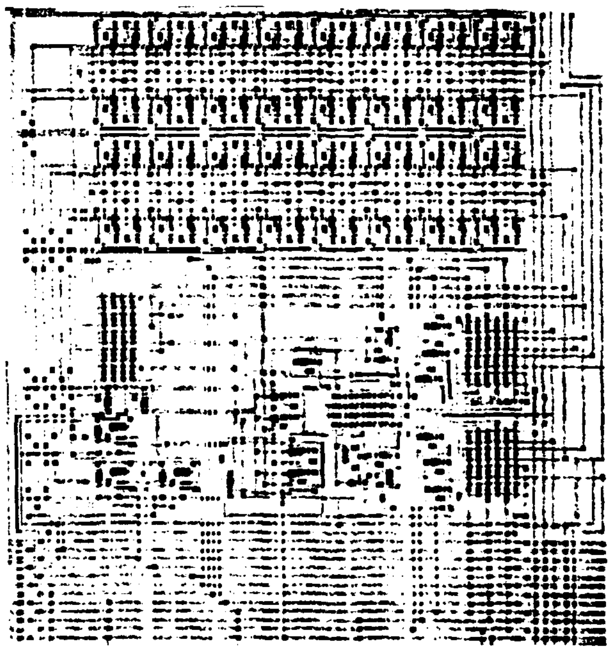


Figure 2. Processing element implementation.

The Grinnell image processor is an option for the Grinnell GMR image display system. Our Grinnell at Stanford has four channels of 8-bit memory configured as 512 × 512 displays, and four 1-bit 512 × 512 "overlay" channels. The image processor is a 16-bit ALU that takes operands from four 8-in 8-out lookup tables. Lockup table inputs can be selected from any of the image memories, and their outputs can be complemented or set to "all zeroes" or "all ones". The ALU can perform a number of arithmetic and logical operations, and its 16-bit output can be written into different image memories (with some minor restrictions). In addition, overlay channels can be written from several 1-bit sources (including the MSB and the carry out of the ALU). Two sets of control registers provide the ability to use an overlay channel to select between different operations on a pixel-by-pixel basis. Since memory operations are synchronized to the display, the image processor finishes an operation in one frame time. In general, this image processor is fairly fast and flexible for integer operations on the image memories, and more detail about its operation can be found in [Grinnell 80].

Another option on the Grinnell, the zoom/pan card, allows an image to be shifted spatially before being used by the image processor. Using this option allows programming local operations similar to those that are well-suited for the processor array discussed in the last section.

We have programmed the Grinnell to implement the edge-magnitude and edge-thinning steps of the Nevatia-Babu line-finding algorithm described in [Nevatia 78]. The edge-magnitude step convolves the original image with six step-edge masks at 30 degree incremental rotations, then finds the direction and value of the convolution with the largest absolute value at each pixel. The edge-thinning step then selects pixels that are flanked (in the direction orthogonal to the edge direction) by pixels which have a smaller edge magnitude and an edge direction within 30 degree of the central pixel. These pixels are labeled edges, and they suppress the labeling of the two pixels in the direction orthogonal to their edge direction (this results in single-pixel width lines). Figure 3 shows an image and the resulting thinned edges (these images were taken directly from the Grinnell system).

One of the things that this experiment shows is the value of specialized systems for computer vision algorithms like this one. By taking advantage of a large memory and a regular addressing scheme, the Grinnell completes these two steps of

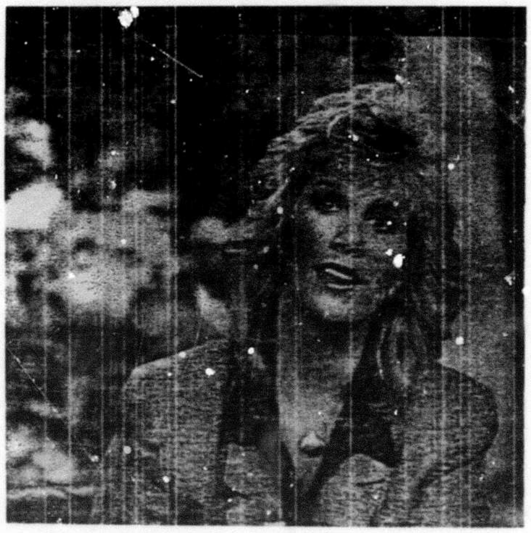
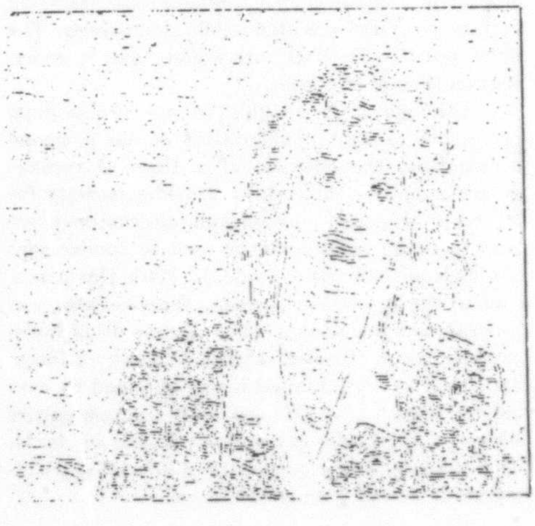


Figure 3. Image and result of Nevatia-Babu algorithm.



the Nevatia-Babu algorithm in under 15 seconds. Our experience shows that the same algorithm (the original version from University of Southern California) takes about 30 minutes to run on the same size image using a lightly-loaded timeshared KL-10. This speedup of two decimal orders of magnitude has prompted us to begin implementation of other algorithms in use at Stanford on the Grinnell system, and is a good indication of the utility of further research on our processor array.

Simulator

To better understand the limitations of the processor array in implementing algorithms and to get some specific data on the running times of these algorithms, we have implemented a simulator of the processor array. The simulator has proved useful in suggesting design modifications and extensions to the original processing element to make it easier to program and faster in executing vision algorithms.

Since the original processing element was designed to support an unsigned convolution, this algorithm is a natural first step in simulation. Figure 4 shows a segment of a line and the result of convolving it with a 3×3 mask with all mask entries set to 1. (This process is a rather crude method for filtering lines to display them on a multi-bit display device). The convolution was simulated with the original processing element configuration described in [Lowry 81] and executed in 565 clock cycles. The estimated 10 MHz clock rate would result in an execution time of 56.5 μsec.

Although it is possible to use the original processing element configuration to do a signed convolution, the addition of a third D register greatly simplifies the task by providing storage for a precomplemented multiplicand (alternatively, extra processing steps could be used to complement the multiplicand when needed). With this extension to the processing element, the simulator was programmed to do a signed convolution using two's-complement arithmetic. Figure 5 shows an image and the laterally inhibited image obtained by convolving it with a 5×5 mask with all mask entries set to -1 except for the center which is set to 24. The simulated convolution takes 1.57 clocks, yielding an execution time of 1.36 msec.

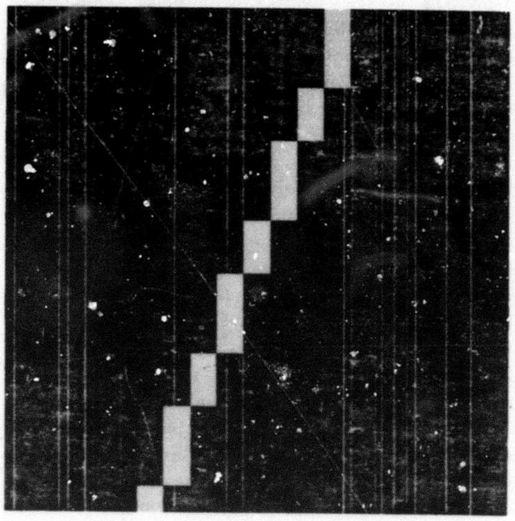
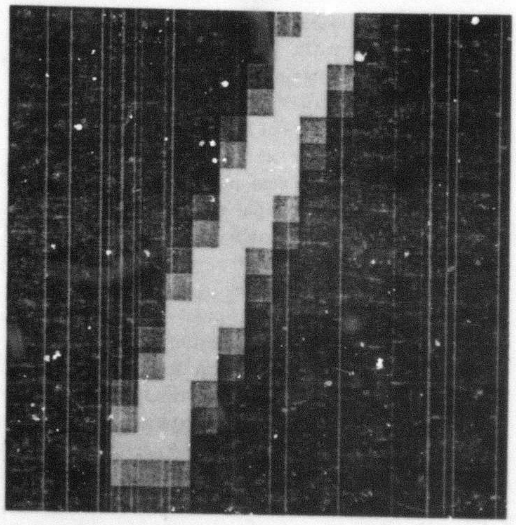


Figure 4. Image and result of unsigned convolution.



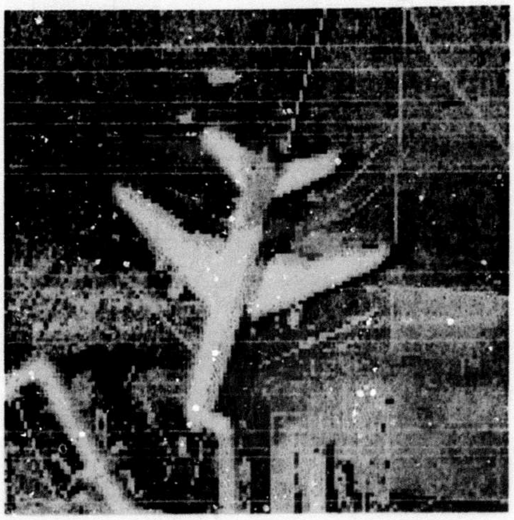
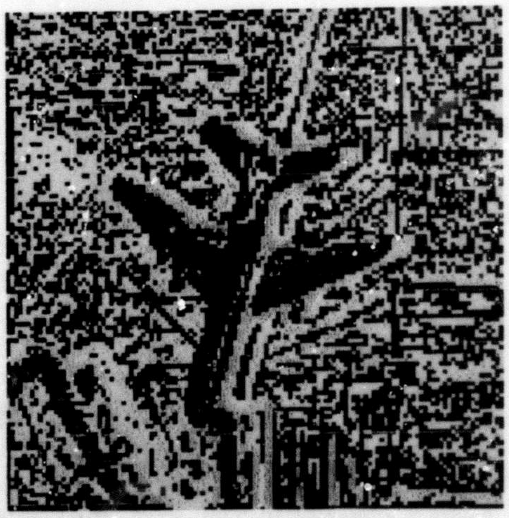


Figure 5. Image and result of signed convolution.



Fault-tolerance

Integrated circuit yield decreases exponentially with area. In order to make a wafer-scale integrated circuit fault-tolerant, it is necessary to incorporate redundant elements and reconfigurable interconnections. For a static array where the interprocessor communication overhead needs to be kept low, non-volatile hardware is the appropriate choice for reconfiguration. We are currently considering laser-fusible links, the re-routing technology being developed by the restructurable VLSI project at Lincoln Laboratories. [Blankenship 82] describes this project including a routing algorithm for embedding a set of one-dimensional 2-neighbor connected arrays into a 2-dimensional physical array with defects. In this report we describe a routing algorithm that embeds a logical 2-dimensional array with 4-neighbor connectivity into a physical 2-dimensional array with 8-neighbor connectivity.

The effectiveness of routing algorithms to successfully reconfigure a chip with defective elements depends both on the defect rate and the flexibility of the interconnection scheme. Although data on defect rates is not easily available from most manufacturers, we can roughly estimate the defect rate for an individual processing element by comparing its size to the area of other chips and using the exponential model of yield. Based upon a conservative comparison, we expect defect rates in processing elements to be between 5% and 10%. A reconfiguration scheme based on disconnecting entire columns containing a defective processing element would be unacceptable for large arrays. For example, only 0.14% of the columns in a 128×128 array would be defect-free with a processing element defect rate of 5%. A routing algorithm that operates at the level of individual processing elements rather than large groups of processing elements is needed.

Routing Algorithm

The routing scheme we developed maps an $N \times N$ logical array onto an $M \times N$ physical array of processing elements, where $M > N$. Columns in the logical array are mapped onto columns in the physical array, while rows in the logical array are mapped onto paths that follow diagonal and horizontal links in the physical array. Defective processing elements are skipped along columns, while they are detoured along rows. Figures 6 and 7 show the redundant links for connections along rows and columns respectively, with marked locations indicating where a laser would fuse unwanted links. Since physical columns correspond to logical columns, only extra rows are needed. Figure 8 shows a 10 × 10 logical array mapped onto a physical array with a 20% defect rate using 5 extra rows. The algorithm starts with the bottom row in the logical array and finds a route for each successive row in the logical array. For each row, a route is found from left to right using the diagonal and horizontal links, keeping as close to the bottom of the physical array as possible. If a dead end is reached while routing a row the algorithm backtracks along the row. Once a row is routed, it stays fixed. Each processing element makes a connection to its logical right neighbor by attempting physical connections to the lower right, right, or upper right, in that order. A connection fails if the processing element being connected to is already assigned, is defective, or leads to a previously determined dead end. The algorithm is linear in the number of processing elements since each element is backtracked over at most once. The running time on a KL-10 for a 128 × 128 array with a 20% defect rate is 1.5 seconds.

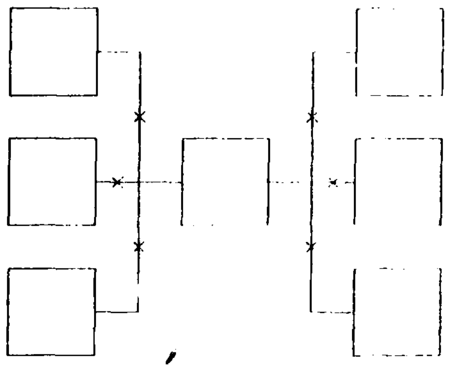


Figure 6. Redundant links along rows.

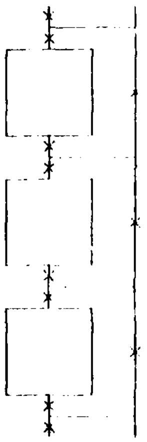


Figure 7. Redundant links along columns.

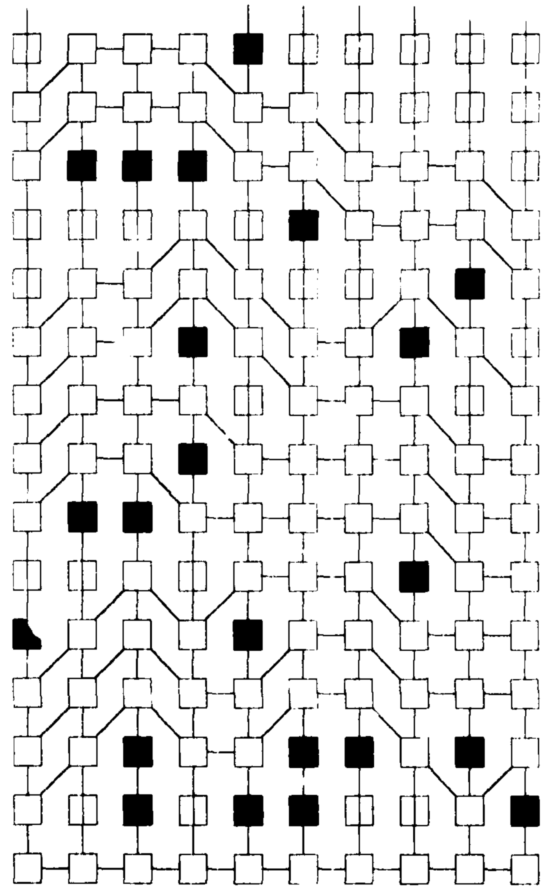


Figure 8. Rerouted 10 × 10 array. Black boxes are defective processors. Boxes with vertical bars through them are unused.

Defect rate	Logical array size 16	32	64	128	256
0.01	1.13/1.13	1.09/1.09	1.06/1.06	1.05/1.05	1.04/1.04
0.02	1.19/1.13	1.13/1.13	1.09/1.09	1.09/1.07	1.07/1.06
0.05	1.25/1.25	1.21/1.19	1.20/1.16	1.17/1.13	1.16/1.11
0.10	1.50/1.38	1.40/1.31	1.37/1.27	1.35/1.22	1.34/1.21
0.20	2.00/1.69	1.96/1.56	1.93/1.48	1.91/1.42	1.88/1.38
0.25	2.38/1.81	2.44/1.72	2.39/1.61	2.38/1.54	2.34/1.48
0.30	2.94/2.00	3.12/2.25	3.11/2.09	3.13/1.98	>2.00/1.61
0.40	5.19/2.44	5.94/2.25	6.96/2.09	>4.00/1.98	>2.00/1.90
0.50	>8.00/3.06	>8.00/2.78	>8.00/2.58	>4.00/2.43	>2.00/2.32

Figure 9. The ratio of total rows needed / rows routed to insure that 90% of the wafers can be successfully routed. This ratio is a function of the defect rate for individual processing elements and the size of the $N \times N$ array that is logically implemented. Each entry consists of the empirically derived ratio on the left and the theoretical lower bound on the right.

Analysis of Routing Algorithm

A lower bound on the number of extra rows needed to route an $N \times N$ logical array is the expected maximum number of defects along columns in the physical array. The probability distribution for this function is a cumulative binomial distribution raised exponentially to the number of columns. The exponential causes the probability distribution to asymptotically approach a vertical step edge. Thus there is little difference between the number of rows needed to insure that 25% of the wafers can be successfully routed and the number needed for 90% successful routing. This probability function was used in computing theoretical lower bounds on the number of redundant rows needed to insure 90% successful routing of a wafer as a function of the processing element defect rate and the size of the logical array. Figure 9 compares these lower bounds with results from Monte Carlo simulations.

Thousands of simulations were run to verify the viability of the routing algorithm. For defect rates of less than 10% the empirical results are in excellent agreement with the theoretical lower bound. For defect rates greater than 20% the empirical results become increasingly larger than the theoretical lower bound. It would appear that the number of dead end routes becomes the dominant factor with large defect rates The results show that for a fixed defect rate the number of extra rows needed to insure 90% routability is a decreasing percentage of the linear dimension of the logical array. This percentage is roughly 16% for a 5% defect rate, 34% for a 10% defect rate, and 88% for a 20% defect rate. Thus this routing scheme appears viable for defect rates up to 10%, and performance increases with larger array dimensions.

Evaluation of Routing Scheme

The routing scheme has the advantages that only extra rows are needed, the reconnections are relatively simple compared to those required for arbitrary reconnections between neighbors, and it is compatible with laser technology being developed for restructuring wafer-scale VLSI chips. The algorithm is simple, efficient,. and linear in the number of processing elements, even though the general problem of finding a subgraph isomorphism is NP-complete. [Garey 79]

For defect rates higher than 10%, this scheme could be extended to allow skipping along rows and routing vertical links along diagonals. Extra rows and columns would be needed, but the scheme would no longer be limited by the maximum number of defects along a column. A more sophisticated routing algorithm would be used.

Although the algorithm was designed and tested assuming no defects in the interconnection hardware, simple modifications would enable it to route around bad interconnections. Defects in the horizontal and diagonal interconnections would be incorporated in the test to connect to a right neighbor. Defects in interconnections between vertically adjacent neighbors would be handled by treating one of the adjacent processing elements as defective. Defects in the interconnections that skip defective processing elements along columns cannot be tolerated; however, considerable redundancy can be incorporated into these interconnections by fabricating two lines and connecting them at each processing element with a fusible link. Any break in one line will then be compensated for by the other line.

Summary

In this paper we reviewed an architecture for a computer vision rectangular processor array that is suitable for VLSI implementation. The results of an array simulator applied to convolutions show that the speed and internal memory of the bit-serial processing elements is well suited to vision algorithms. Rough calculations indicate a defect rate between 5% and 10% for individual processing elements. A routing scheme suitable for these defect rates and large arrays was implemented and subjected to extensive simulation. The results indicate the feasibility of implementation on a wafer-scale chip using technology being developed for restructurable VLSI.

This work was supported by ARPA contract N00039-82-C-0250

References

[Blankenship 82] Blankenship, P.E. "Restructurable VLSI Program — SemiAnnual Technical Summary Report to DARPA," Lexington, Mass, January 1982.

[Duff 81] Duff, M. J. B., and S. Levialdi, **Languages and Architectures for Image Processing,** Academic Press, London, 1981.

[Garey 79] Garey, Michael R., and David S. Johnson, **Computers and Intractability — a Guide to the Theory of NP-Completeness,** W. H. Freeman, San Francisco, 1979.

[Grinnell 80] Grinnell Systems Corporation, *GMR 270 User Manual,* San Jose, 1980.

[Lowry 81] Lowry, Michael R., and Allan Miller, "A General Purpose VLSI Chip for Computer Vision with Fault-tolerant Hardware," *Proc. ARPA Image Understanding Workshop,* April 1981.

[Mead 80] Mead, Carver, and Lynn Conway, **Introduction to VLSI Systems,** Addison-Wesley, Reading, 1980.

[Nevatia 78] Nevatia, R., and K. R. Babu, "Linear Feature Extraction," *Proc. ARPA Image Understanding Workshop,* Pittsburgh, November 1978.

P000140

SEGMENTATION OF FLIR IMAGES: A COMPARATIVE STUDY

Ralph L. Hartley
Leslie J. Kitchen

Cheng-Ye Wang
Azriel Rosenfeld

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

Several segmentation techniques were applied to a set of 51 FLIR (Forward-Looking InfraRed) images of four different types, and the results were compared to hand segmentations. There were substantial differences in performance, indicating that the choice of proper technique is very important. The segmentation techniques used were "superslice", "pyramid spot detection", two versions of "relaxation", "pyramid linking", and "superspike". One technique, "superspike", outperformed all the others, detecting 88% of the targets and yielding only 1.6 false alarms per true target.

1. Introduction

Object detection in infrared images is a problem of considerable practical interest [1]. Numerous techniques have been developed for the primary purpose of segmenting FLIR (=Forward Looking InfraRed) images into objects and background (e.g., [1,2]); in particular, [3] is a survey of such techniques, and [4] describes a comparative study. This paper summarizes the results of another comparative study; further details about the study can be found in [5].

Section 2 describes the segmentation techniques that were tested; Section 3 describes the evaluation procedure; and Section 4 summarizes the results of the study.

2. Segmentation techniques

The techniques tested are briefly described in the following paragraphs; for further details see the cited references.

2.1 Superslice [6]

This technique was quite successful in earlier studies of FLIR object detection [1]. A set of gray level thresholds is applied to the given image, and for each threshold, connected components of above-threshold points are extracted. The gray level gradient is also measured for the image, and points at which it is a local maximum are determined. A component is selected as a possible object if many gradient maxima coincide with its border and surround it.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Clara Robertson in preparing this paper.

2.2 Pyramid spot detection [7]

This technique is designed to extract compact objects of arbitrary size from an image; it too performed well in earlier studies. We build an exponentially tapering "pyramid" of reduced-resolution versions of the image by successive block averaging, e.g., using nonoverlapping 2x2 blocks, or 4x4 blocks with 50% overlap in each direction, so that each image is half the size (¼ the area) of the preceding. At each level of the pyramid, we apply a standard spot-detection operator - e.g., we compare each pixel to its eight neighbors, and judge a spot to be present if they differ sufficiently. A spot that is detected in this way should correspond to a compact object on a contrasting background in the original image. For each such spot, we consider the portion of the original image corresponding to the pixel and its neighbors, and apply a threshold to this portion, chosen midway between the gray level of the pixel (an average of a block of gray levels in the original image) and the average gray level of its neighbors (an average of block averages). This thresholding generally extracts the object that gave rise to the spot detection.

2.3 Relaxation [8]

"Relaxation" methods of object extraction have been extensively studied. The basic approach is to initially assign "object" and "background" probabilities to each pixel, based on their distances from the ends of the grayscale. The probabilities are then iteratively adjusted based on the probabilities of the neighboring pixels, with like reinforcing like. When this is done, the probabilities tend to converge to relative certainty ((0,1) or (1,0)), and yield a good segmentation of the image into objects and background. An alternative, also investigated, used three rather than two classes, assigning initial probabilities based on distances from the ends and midpoint of the grayscale; thus the pixels were not forced to choose between "target" and "background", but also had a third option ("clutter").

2.4 Pyramid linking [9]

This is a method of segmenting an image based on creating links between pixels at successive levels of a "pyramid". We build the pyramid using overlapping 4x4 blocks; thus each pixel has 16 "sons" (on the level below) that contribute to its average, and four "fathers" (on the level above) to whose average it contributes. We now link each pixel to the father whose value (=average) is closest to its own. We then recompute the averages, allowing only those sons that are linked to a pixel to contribute to its average. We now change the links based on these new averages, then recompute the averages again, and so on. This process stabilizes after a few iterations; at this stage the links define subtrees of the pyramid, rooted at the top level, which we take to be 2x2, so that there are (at most) four trees. The sets of leaves of these trees (pixels in the original image) thus define a segmentation of the original image into at most four subsets.

2.5 "Superspike" [10]

This is a method of image smoothing based on iterated selective local averaging. Each pixel is averaged with those of its neighbors that satisfy the following criteria, based on the image's histogram:

a) The neighbor is more probable than the pixel, i.e., its gray level has a higher value in the histogram.

b) The histogram has no concavity between the gray levels of the pixel and the neighbor (as would be the case if they belonged to two different peaks, or to a peak and shoulder).

When this process is iterated a few times, the histogram generally turns into a small set of spikes. The image can then be segmented by mapping them into nearby taller ones, until only five spikes remained, thus segmenting the image into five subsets. The choice of five classes was an arbitrary one, based on preliminary experiments in which it was found that using fewer classes tended to merge some objects into the background.

3. Methodology

The overall approach used in the comparative study was as follows:

1) Each technique being tested (Section 2) was applied to the given set of images, yielding a classification of each image into subsets. Connected component labelling was performed on the resulting classified images, yielding a set of regions.

2) Regions that were too large, too small, or too elongated to be targets were eliminated. In our main study, the criteria for acceptability were

$$\left.\begin{array}{l} 4 \le \text{height} \le 41 \\ 3 \le \text{width} \le 50 \end{array}\right\} \quad \text{(pixels)}$$

$$0.4 \le \text{aspect ratio} \le 2.0$$

In addition, regions having the wrong polarity relative to the mean image gray level were eliminated.

3) For each surviving region, the coordinates of its centroid and the dimensions of its upright circumscribing rectangle were computed. The centroids and circumrectangles of the true targets were also known (from ground truth information and hand segmentation). A target was said to have been detected if the x and y displacements between a region centroid and a true target centroid were at most half the true target's rectangle dimensions. Region centroids not satisfying these conditions were considered to be false alarms. The "segmentation accuracy" for each detected target was measured by the fraction of overlap between the circumrectangle of the detected region and that of the true target. "Extra detections" were said to occur when more than one region centroid occurred in the inner half of a true target's rectangle; all such detections were counted in computing the average segmentation accuracy. These methods of evaluating a segmentation were proposed in [3].

4. Experiments

In a pilot study, all six techniques (including both two-class and three-class relaxation) were applied to three image samples (see Figure 1). Figure 1 also shows the resulting segmented images. We see that the pyramid spot technique did not perform very well. This is not too surprising, since this technique was designed for the extraction of isolated objects on a contrasting background. Results with the relaxation, pyramid linking, and superspike techniques looked more promising, and it was therefore decided to use all of them in the main study. The superslice technique was not used in the main study because of its comparatively high computational cost, which made its use relatively impractical.

The main study used a set of 51 FLIR images supplied by Westinghouse Systems Development Division [3] from Navy (Nos. 2-10), Army (Nos. 11-30, 55-70), and Air Force (Nos. 31-36) sources (Figure 2).* All images are 128x128; Nos. 11-30 were obtained from 64x64 images by horizontal and vertical reflection, in order to present the targets in four orientations.

* **Further information about the data base can be obtained from Mr. Bruce J. Schacter, Westinghouse Systems Development Division, Baltimore, MD 21203. The target types and locations are listed in Table 1.**

The four selected techniques (two- and three-class relaxation, pyramid linking, and superspike) were applied to these images. [In the case of images 21-30, they were applied to only one quadrant, since the methods are essentially orientation-invariant; the scores (detections and false alarms) obtained in this way were multiplied by 4.] The pyramid linking algorithm was designed for 64x64 images;* in order to apply it to images 2-10, 31-36, and 55-70, they were resampled down to that size, and the outputs (centroids and rectangles) were scaled up in order to compare them with the ground truth.

Figure 3 shows the segmentation results using the four methods for each of the 51 images. Table 2 summarizes, by image class, the number of targets present, the number correctly detected, the number of extra detections, the number of false alarms, and the segmentation accuracy. Detailed results for the 51 individual images are given in [5].

We see from these results that segmentation accuracy does not vary greatly among the methods; it ranges between about .5 and .8 in all cases. Extra detections are also not a significant factor, except perhaps for the pyramid linking and superspike methods as applied to the NVL data (images 11-30). As regards correct detections and false alarms, 3-class relaxation and superspike were the best methods (though no method was very good) for the Navy images; pyramid linking and superspike had good detection rates for the NVL data, but the former had a much higher false alarm rate; and superspike was by far the best method for the Air Force and NVL flight test images, making it the best method overall. It detected 111 of the 126 targets (over 88%) with only 26 extra detections and 202 false alarms (about 1.6 per true target), and its segmentation accuracy was a reasonable 0.66. The next best method, pyramid linking (which, it should be recalled, was applied to half-resolution versions of images nos. 2-10, 31-36, and 55-70), detected only 63% of the targets and had many more false alarms (over 5 per target). For further details see [5].

5. Concluding remarks

The results of the main study show that one method, "superspike", performed substantially better on the Westinghouse data base than the other methods tested. It detected nearly 90% of the true targets and gave only 1.6 false alarms per target. Note that these results were obtained using segmentation alone, in conjunction with very crude size and height:width criteria. If the segmentation step were followed by a classification algorithm, such better performance could be expected.

Some further improvement in performance can undoubtedly be obtained by further refining the segmentation process. However, there are limits to what can be achieved in this way by algorithms that incorporate so little knowledge about the nature of the targets. In order to attai a significantly higher level of performance, it will probably be necessary to develop a knowledge-driven system capable of some degree of reasoning about the regions extracted by the initial segmentation.

*Extension of this algorithm to 128x128 images is straightforward, but would involve excessive memory requirements.

REFERENCES

1. D. L. Milgram and A. Rosenfeld, Object detection in infrared images, in L. Bolc and Z. Kulpa, eds., Digital Image Processing Systems, Springer, New York, 1981, pp.228-353.

2. L. G. Minor and J. Sklansky, The dectection and segmentation of blobs in infrared images, IEEE Trans. Systems, Man, Cybernetics 11, 1981, 194-201.

3. B. J. Schachter, A survey and evaluation of FLIR target detection/segmentation algorithms, Westinghouse Electric Corp. Systems Development Division, Baltimore, MD [October 1981].

4. M. Burton and C. Benning, A comparison of imaging infrared detection algorithms, Proc. SPIE 302, 1981, 1-8.

5. R. L. Hartley, L. J. Kitchen, C. Y. Wang, and A. Rosenfeld, A comparative study of segmentation algorithms for FLIR images, TR-1104, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, September 1981.

6. D. L. Milgram, Region extraction using convergent evidence, Computer Graphics Image Processing 11, 1979, 1-12.

7. M. Shneier, Using pyramids to define local thresholds for blob detection, TR-808, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, September 1979.

8. R. C. Smith and A. Rosenfeld, Thresholding using relaxation, IEEE Trans. Pattern Analysis Machine Intelligence 3, 1981, 598-606.

9. P. Burt, T. H. Hong, and A. Rosenfeld, Segmentation and estimation of image region properties through cooperative hierarchical computation, IEEE Trans. Systems, Man, Cybernetics 11, 1981, 802-809.

10. K. A. Narayanan and A. Rosenfeld, Image smoothing by local use of global information, IEEE Trans. Systems, Man, Cybernetics 11, 1981, 826-831.

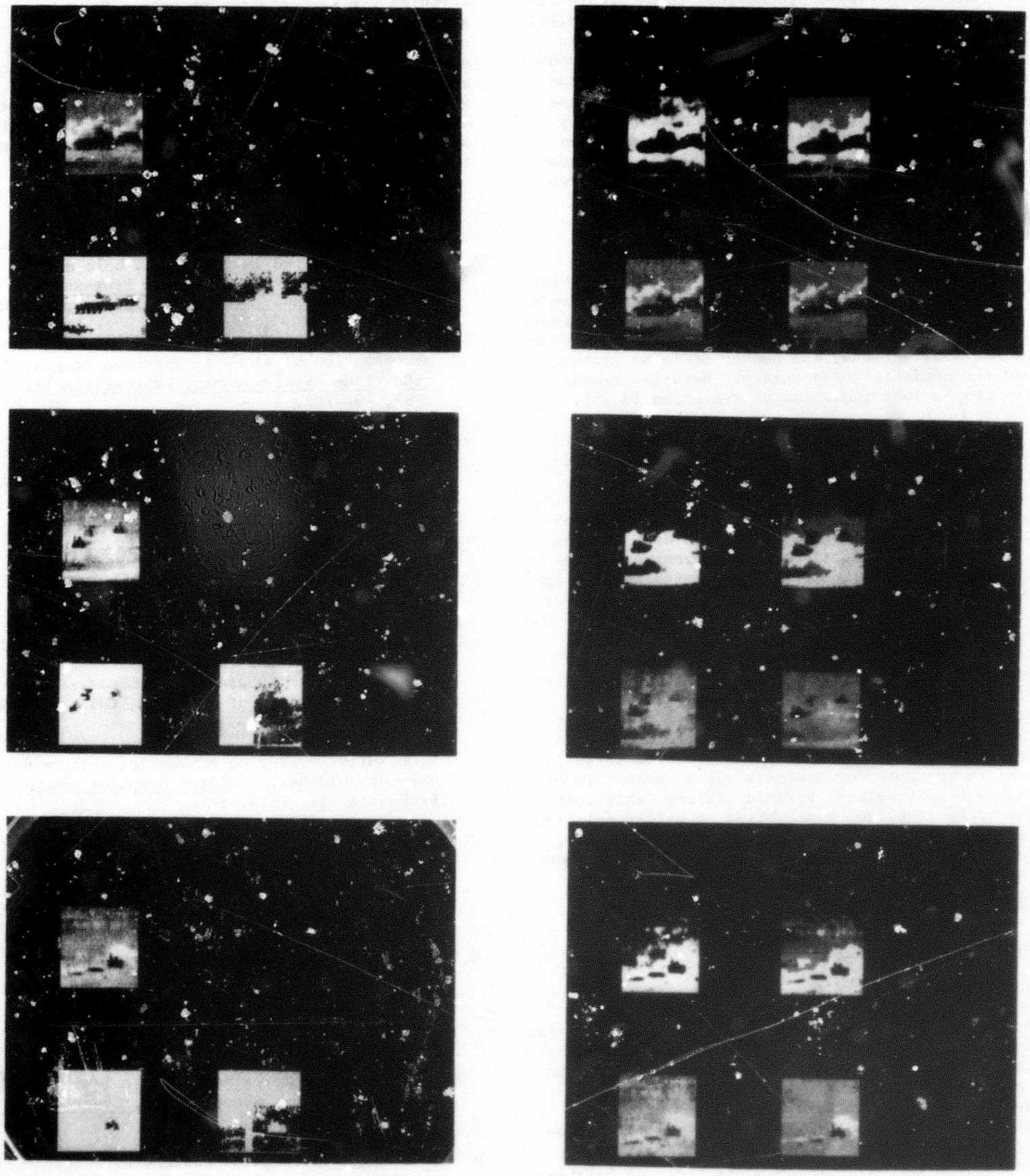


Figure 1. Results of pilot study (three examples).

Input		2-class relaxation	3-class relaxation
Superslice	Pyramid spot detection	Pyramid linking	Superspike

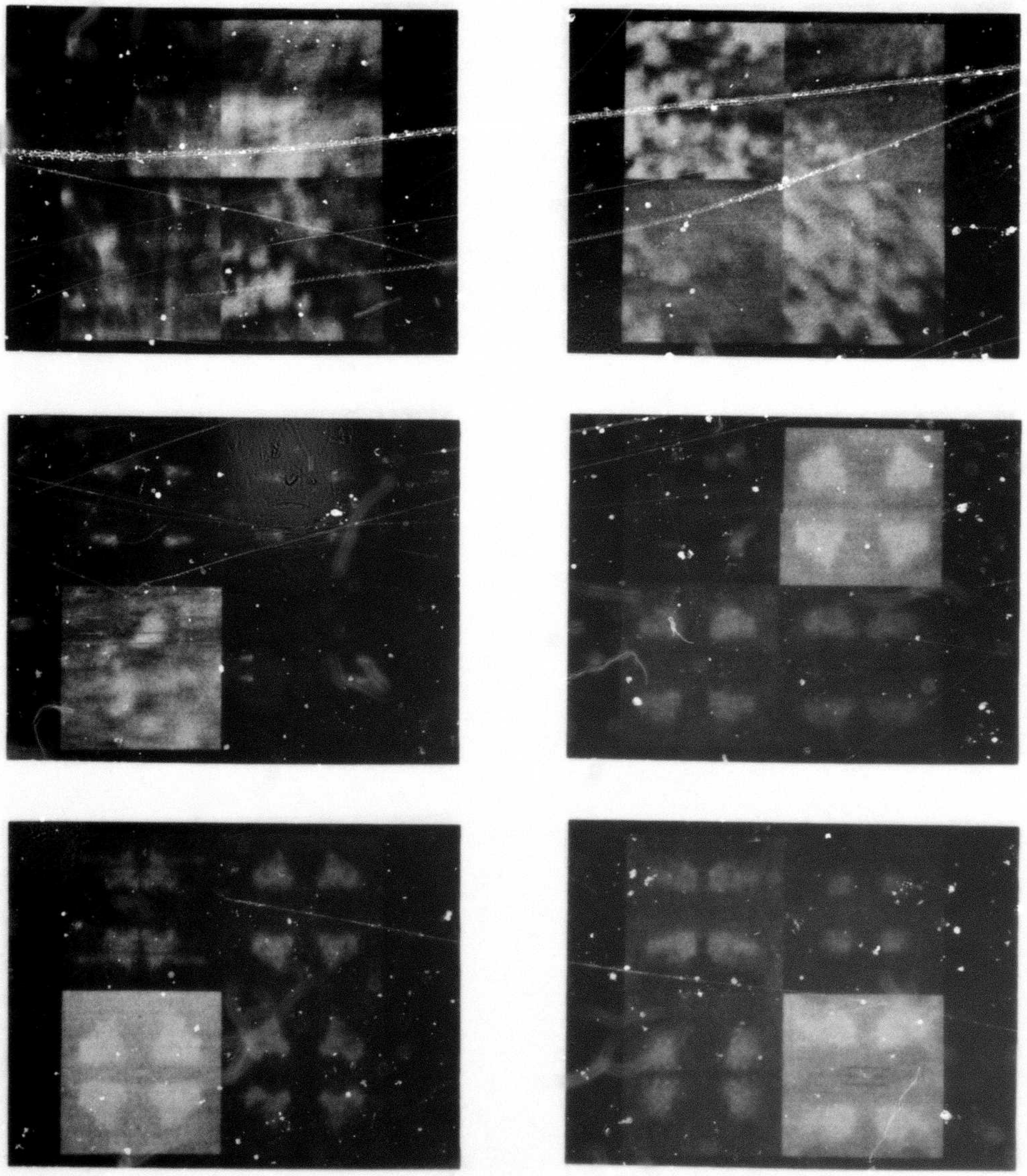


Figure 2. Images used in main study

4 5
2 3

12 13
10 11

20 21
18 19

8 9
6 7

16 17
14 15

24 25
22 23

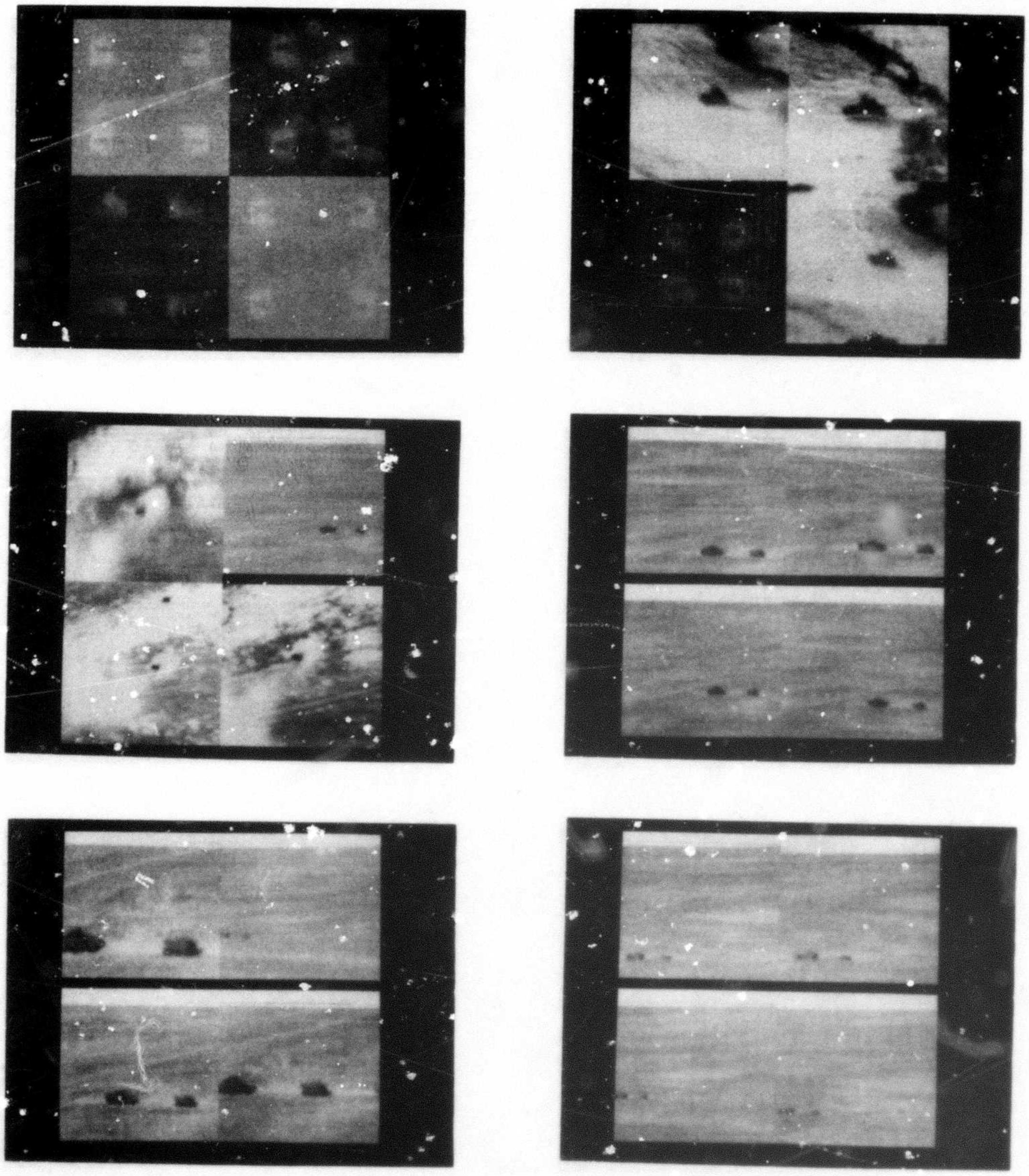


Figure 2. Cont'd.

28 29
26 27

36 55
34 35

62 63
60 61

32 33
30 31

58 59
56 57

66 67
64 65

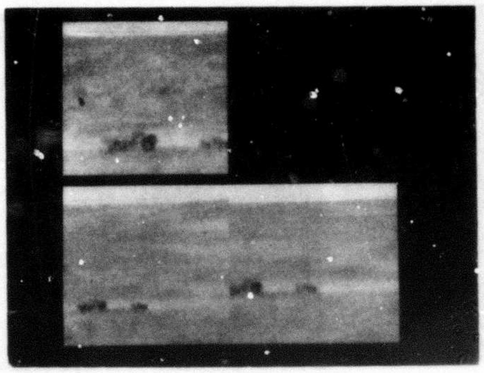


Figure 2. Cont'd.

70
68 69

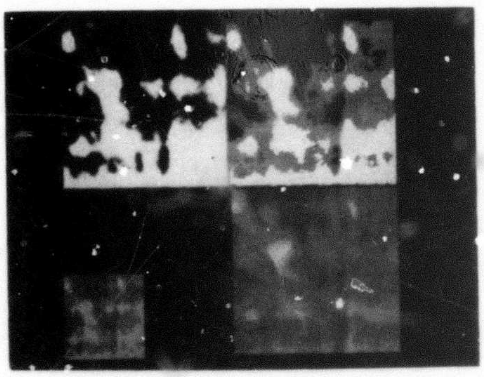
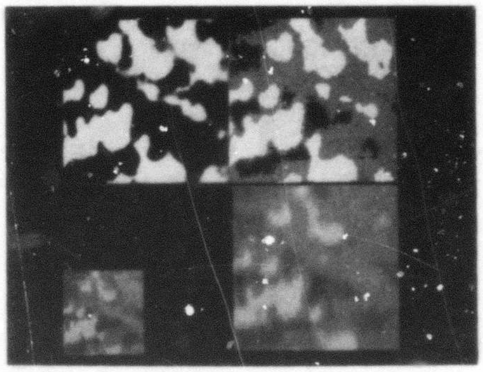
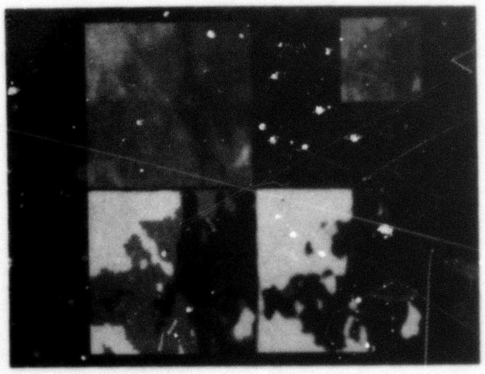
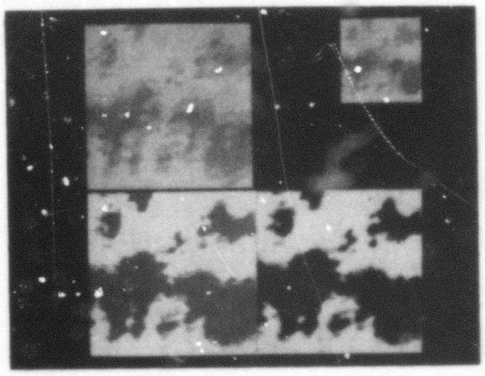





Figure 3. Results of main study

2	3	2-class relaxation	3-class relaxation
4	5	Pyramid linking	Superspike

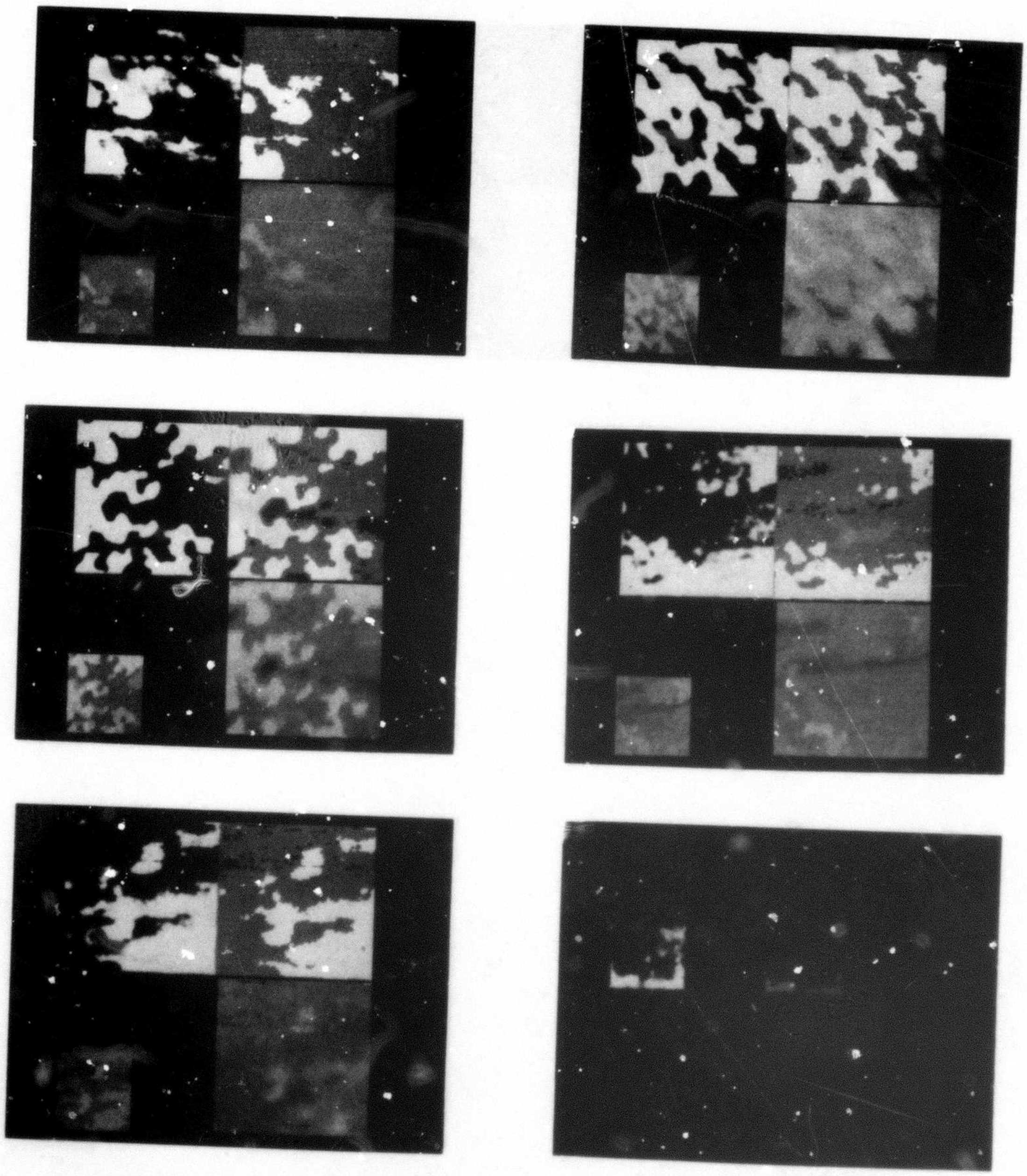


Figure 3. Cont'd.

6 7

8 9

10 11

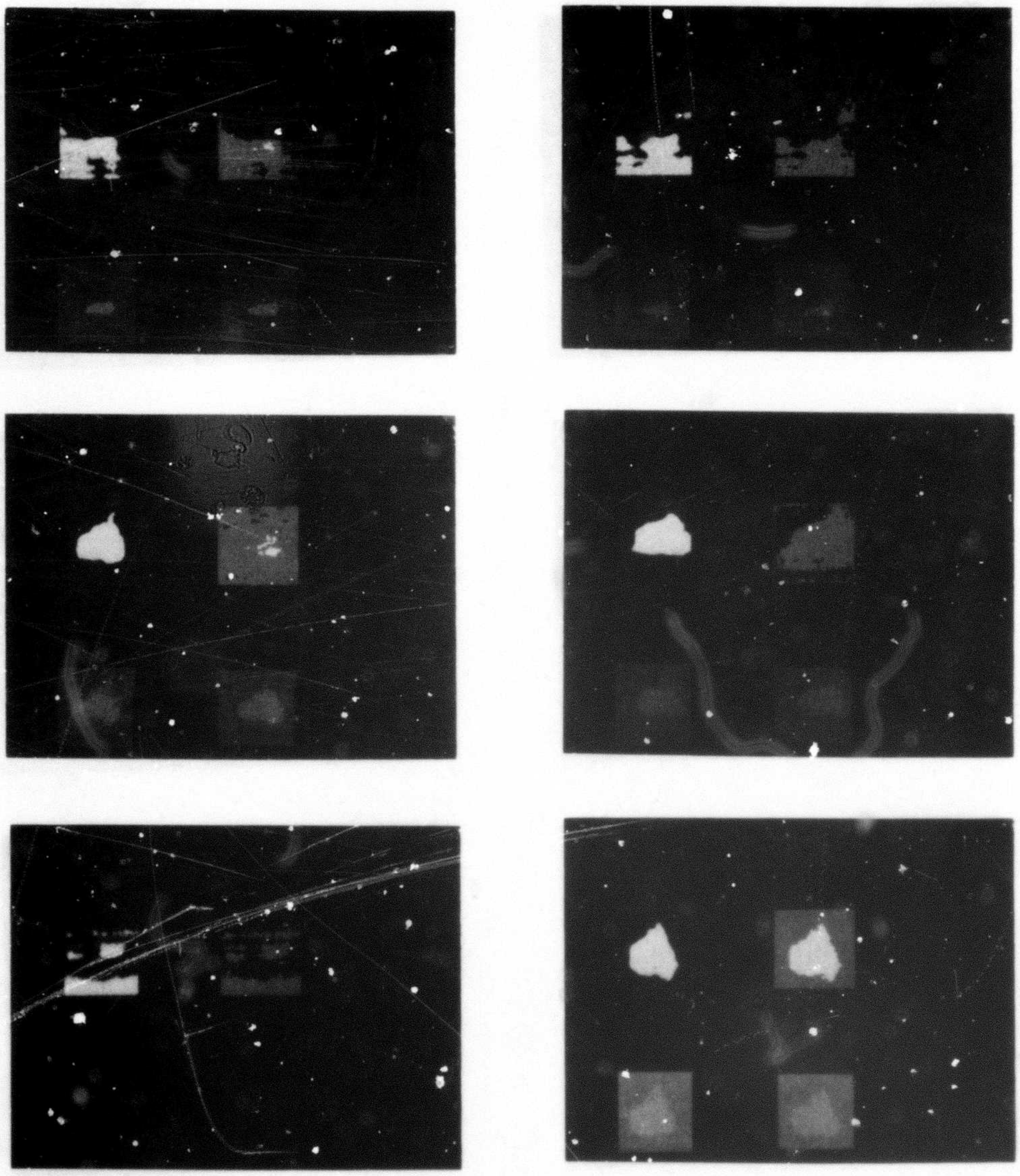


Figure 3. Cont'd.

12 13
14 15
16 17

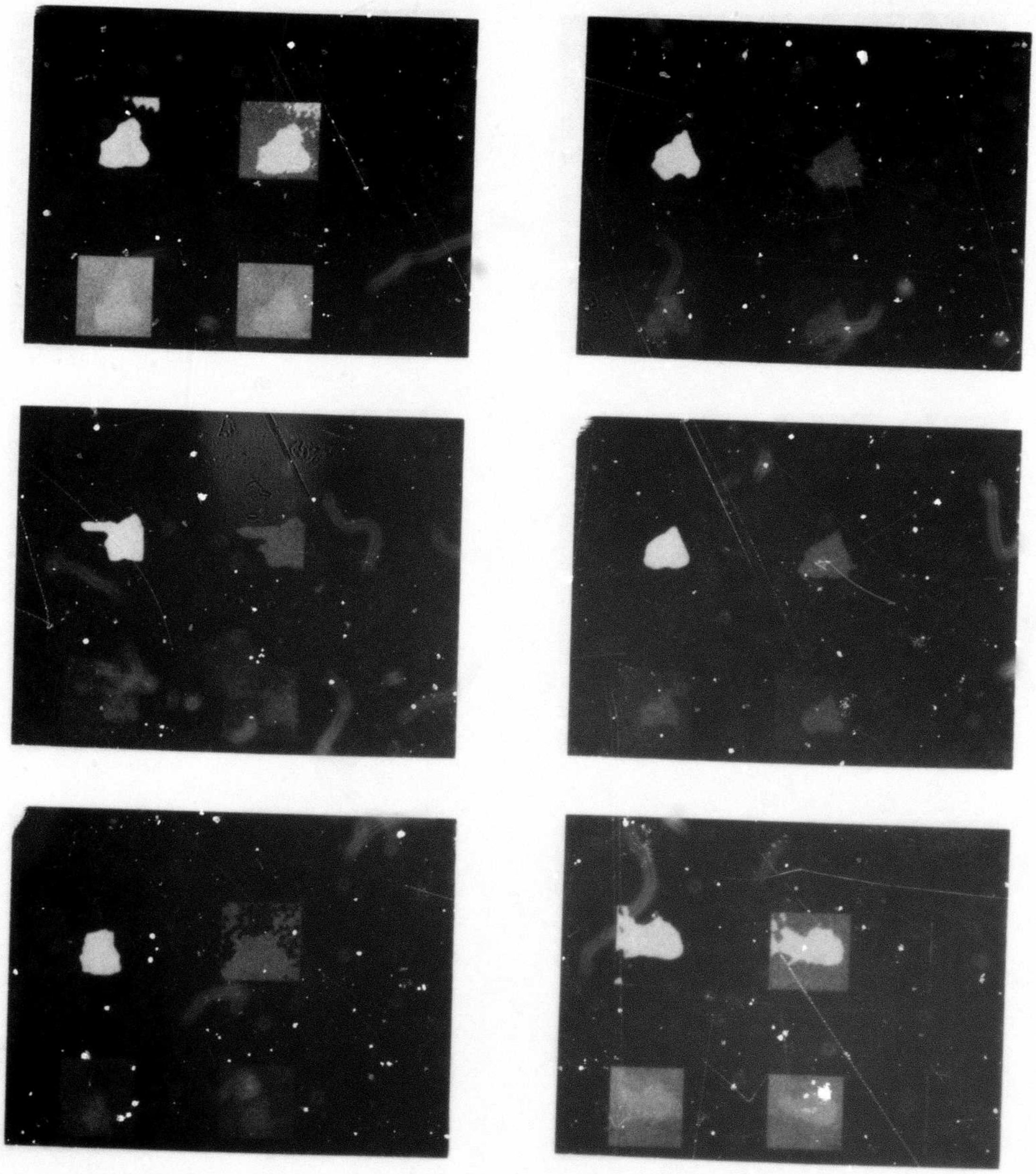


Figure 3. Cont'd.

18	19
20	21
22	23

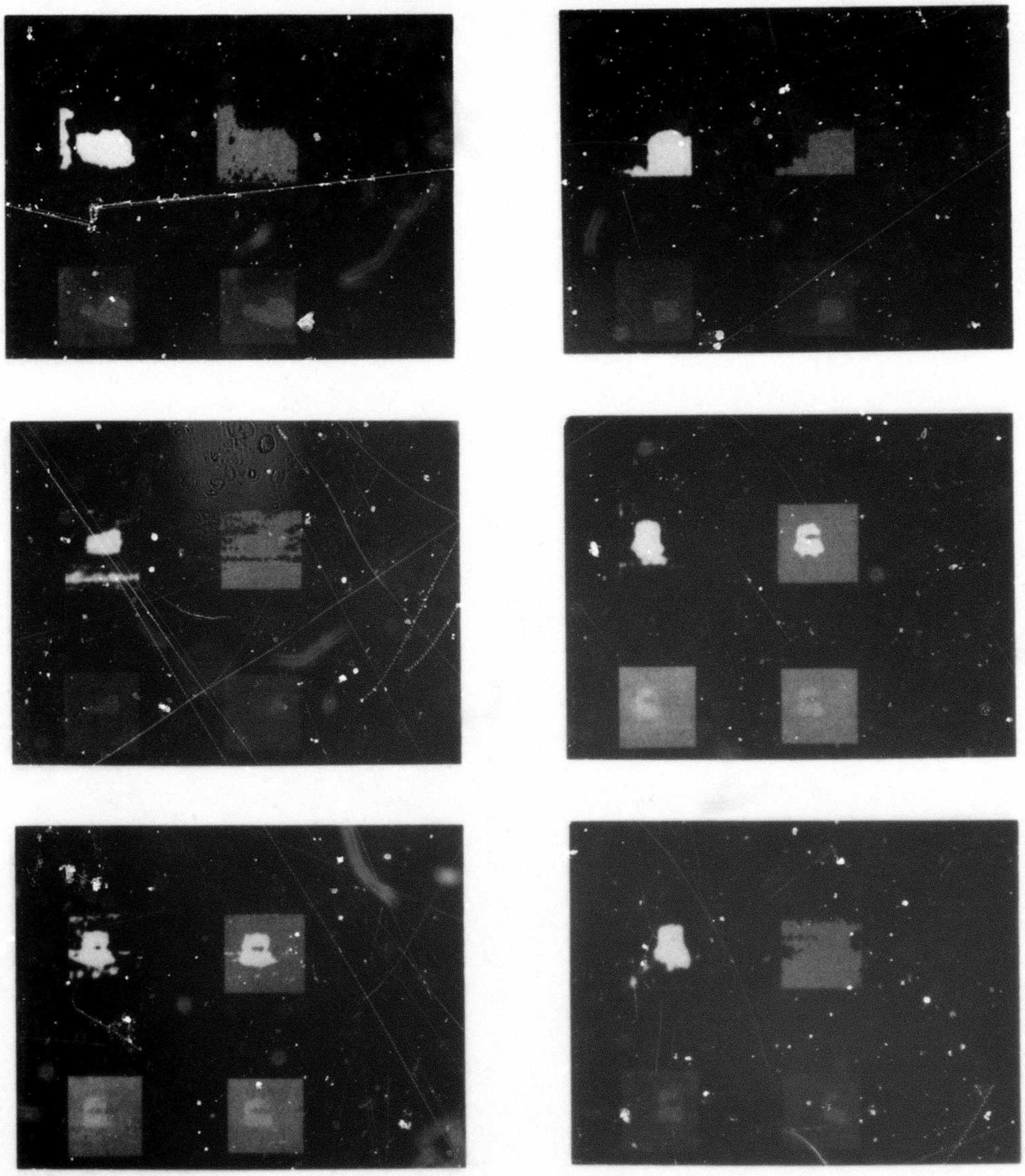


Figure 3. Cont'd.

24	25
26	27
28	29

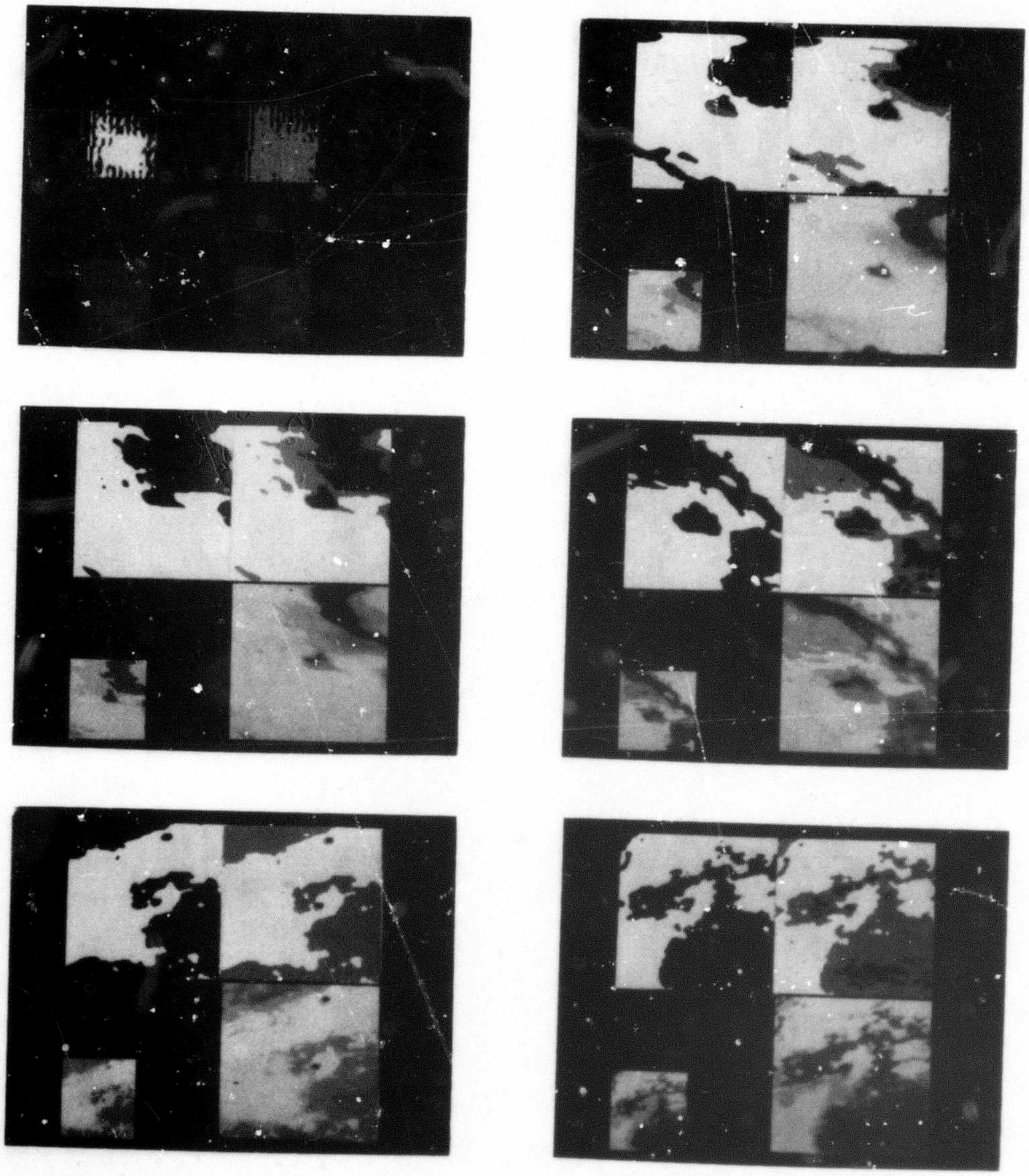


Figure 3. Cont'd.

30 31

32 33

34 35

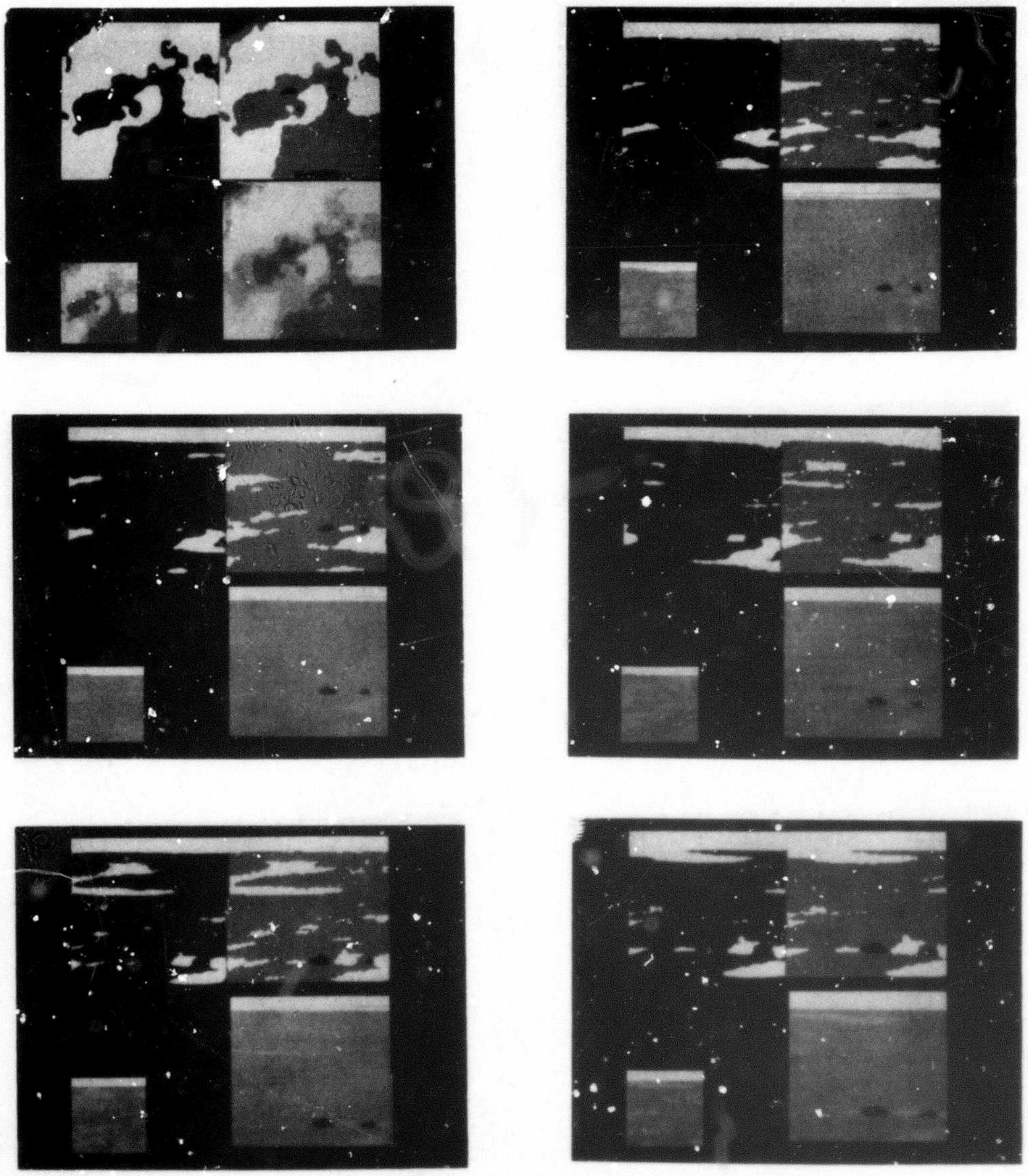


Figure 3. Cont'd.

36	55
56	57
58	59

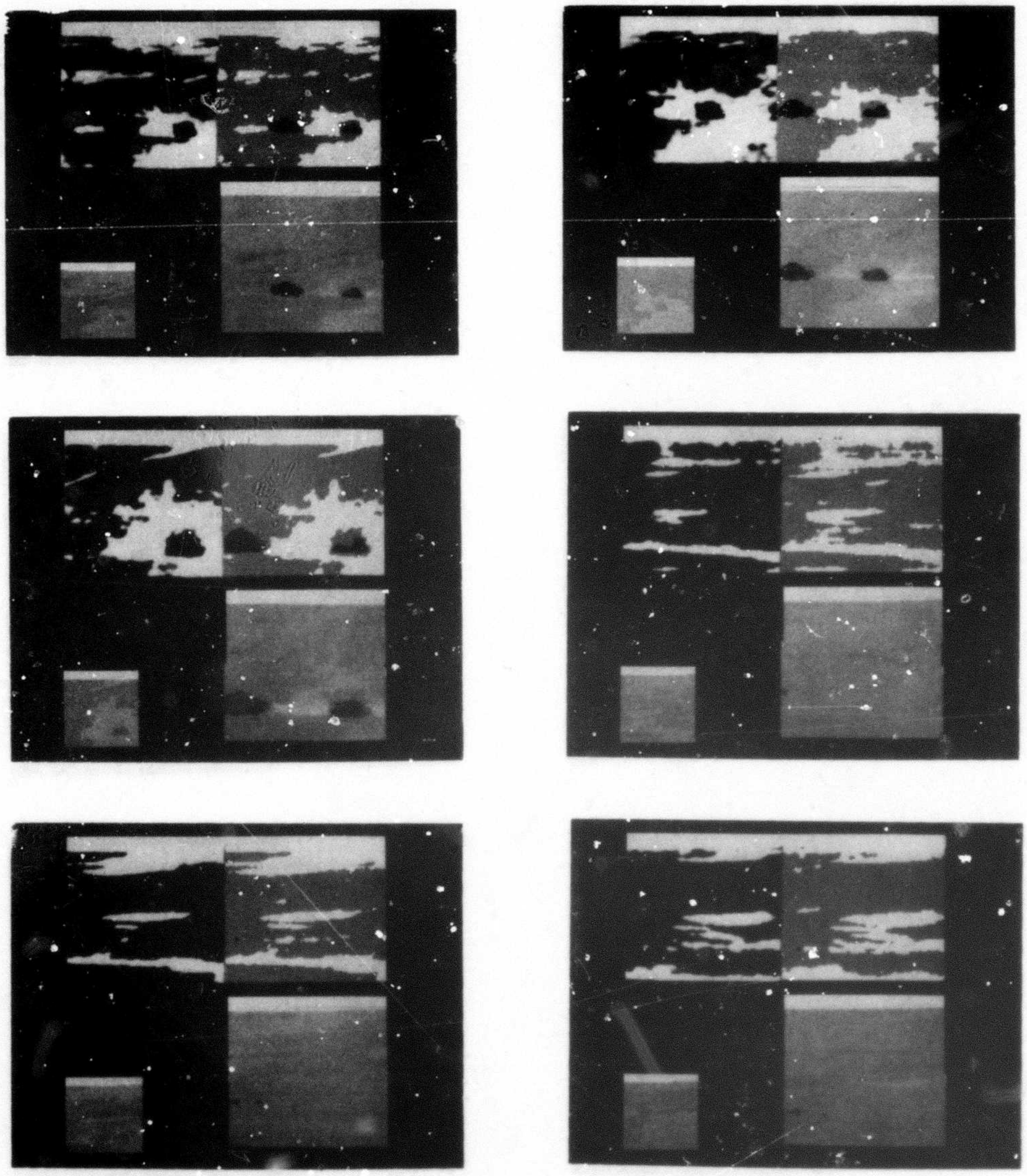


Figure 3. Cont'd.

60	61
62	63
64	65

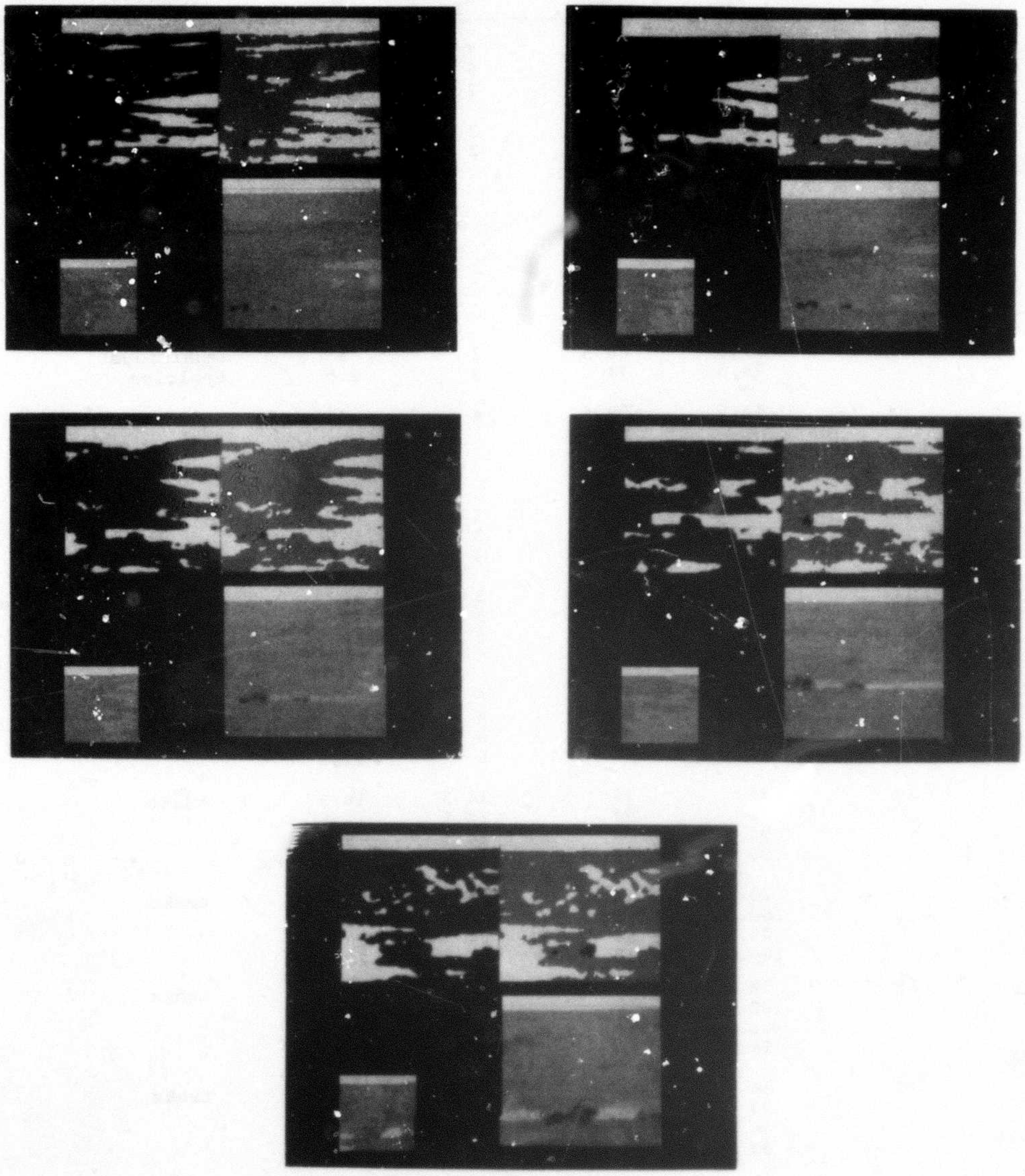


Figure 3. Cont'd.

66 67

68 69

70

Image	C_I	C_J	R_I	R_J	Object(s)
2	19.5	92.5	9	5	trucks
	51.5	91.5	5	4	
	95.5	89	7	3.5	
3	58	67.5	3	2	trucks
	74.5	102	2	1.5	
4	-				none
5	-				none
6	-				none
7	-				none
8	57.5	86.5	2	4	amphibious
	74.5	74	3	2.5	vehicles
9	41.5	92.5	4	4	radar van
10	-				none
11	57.5	19	3	6.5	military
	57.5	110	3	6.5	vehicles
	71.5	19	3	6.5	
	71.5	110	3	6.5	
12	36.5	35.5	5	12	military
	36.5	93.5	5	12	vehicles
	92.5	35.5	5	12	
	92.5	93.5	5	12	
13	40.5	37	6	12.5	military
	40.5	91	6	12.5	vehicles
	88.5	37	6	12.5	
	88.5	91	6	12.5	
14	31	36	14.5	18.5	tanks
	31	93	14.5	18.5	
	98	36	14.5	18.5	
	98	93	14.5	18.5	
15	28.5	40	14	20.5	tanks
	100.5	89	14	20.5	
	28.5	89	14	20.5	
	100.5	40	14	20.5	
16	28.5	42	6	10.5	tanks
	100.5	42	6	10.5	
	100.5	87	6	10.5	
	28.5	87	6	10.5	
17	36	31.5	11.5	17	tanks
	93	31.5	11.5	17	
	36	97.5	11.5	17	
	93	97.5	11.5	17	
18	42	36	13.5	17.5	tanks
	42	93	13.5	17.5	
	87	36	13.5	17.5	
	87	93	13.5	17.5	

Table 1. Ground truth for the 51 images. (C_I,C_J)=centroid coordinates; (R_I,R_J)=half-dimensions of circumrectangle. In images 2-30, high gray levels are hot; in images 31-36 and 55-70, low gray levels are hot.

Image	C_I	C_J	R_I	R_J	Object(s)
19	37	35	18.5	18.5	tanks
	37	94	18.5	18.5	
	92	35	18.5	18.5	
	92	94	18.5	18.5	
20	33	48	17.5	14.5	tanks
	33	81	17.5	14.5	
	96	48	17.5	14.5	
	96	81	17.5	14.5	
21	32.5	39.5	17	18	tanks
	32.5	89.5	17	18	
	96.5	39.5	17	18	
	96.5	89.5	17	18	
22	45.5	31.5	17	17	tanks
	45.5	97.5	17	17	
	83.5	31.5	17	17	
	83.5	97.5	17	17	
23	38.5	34.5	13	23	tanks
	38.5	94.5	13	23	
	90.5	34.5	13	23	
	90.5	94.5	13	23	
24	37	37	13.5	22.5	military vehicles
	37	92	13.5	22.5	
	92	37	13.5	22.5	
	92	92	13.5	22.5	
25	41	46	10.5	13.5	military vehicles
	41	83	10.5	13.5	
	88	46	10.5	13.5	
	88	83	10.5	13.5	
26	27	35.5	9.5	14	jeeps
	102	93.5	9.5	14	
	27	93.5	9.5	14	
	102	35.5	9.5	14	
27	31	25.5	17	14	jeeps
	31	103.5	17	14	
	98	25.5	17	14	
	98	103.5	17	14	
28	29.5	27.5	14	14	jeeps
	29.5	101.5	14	14	
	99.5	27.5	14	14	
	99.5	101.5	14	14	
29	27.5	42	14	11.5	jeeps
	101.5	42	14	11.5	
	27.5	87	14	11.5	
	101.5	87	14	11.5	
30	43	42	9.5	9.5	
	43	87	9.5	9.5	
	86	42	9.5	9.5	
	86	87	9.5	9.5	
31	60.5	75	7	12	tank

Table 1, cont'd.

Image	C_I	C_J	R_I	R_J	Object(s)
32	62	71.5	7.5	14	tank
33	71	59.6	12.5	17	tank
34	64.5	75	11	22.5	tank
35	56	63.5	13.5	19	tank
36	67.5 or 72.5	70 or 62.5	14	15.5	tank
55	83.5 85.5	81.5 105.5	3.5 3	7 6	tank APC
56	84 85	78 106.5	3.5 2.5	7.5 5	tank APC
57	91 93.5	74.5 106.5	3.5 3	8 5	tank APC
58	102 103.5	72.5 108	4.5 3	9 5.5	tank APC
59	96 98.5	68.5 110	4.5 4	11 7.5	tank APC
60	86 88.5	52.5 103	5.5 5	13 8.5	tank APC
61	76.5 78	14.5 76	7 6.5	14.5 11.5	tank APC
62	90 94	18.5 98.5	11.5 8.5	18.5 15	tank APC
63	84.5 84	4 19.5	2 2.5	4 4	truck jeep
64	85 84.5	5 25.5	2.5 3	5 4	truck jeep
65	93.5 95.5	9.5 31.5	3.5 2	7 4	truck jeep
66	102.5 103	14.5 40	4 2.5	8 4.5	jeep truck
67	100 102	24 53	4.5 3.5	10.5 5.5	truck jeep
68	90 91.5	24 61	5.5 3	11.5 6.5	truck jeep
69	78.5 79.5	12 56.5	7 4	12 8	truck jeep
70	96.5 97.5	54.5 118	9.5 4	19 10.5	truck jeep

Table 1, cont'd.

Images	Targets	Method	Correctly detected	Extra detections	False alarms	Segmentation accuracy
2-10 (Navy, China Lake)	8	2-class relaxation	0	0	43	-
		3-class relaxation	2	0	67	0.70
		Pyramid linking	0	0	145	-
		Superspike	3	0	77	0.51
11-30 (NVL data)	80		40	0	92	0.73
			20	8	92	0.49
			72	32	392	0.67
			76	24	60	0.64
31-36 (Air Force, TASVAL)	6		2	0	9	0.74
			3	1	27	0.73
			3	2	100	0.57
			6	1	63	0.60
55-70 (NVL flight test)	32		2	0	6	0.67
			13	1	19	0.65
			4	0	38	0.80
			26	1	2	0.73
Overall	126		44	0	150	0.73
			38	10	205	0.58
			79	34	675	0.68
			111	26	202	0.66

Table 2. Summary of results by image class.

AD-P000 141

THE DARPA/DMA IMAGE UNDERSTANDING TESTBED

Andrew J. Hanson (Testbed Coordinator)
Martin A. Fischler (Principal Investigator)

SRI International, Menlo Park, California 94025

ABSTRACT

The background and current status of the Image Understanding Testbed are described. We give a brief overview of the major software contributions and present selected results of the evaluation process. We also discuss the user programs and user interfaces supported in the Testbed environment and present tentative plans for the future evolution of the Testbed.

I OVERVIEW OF THE IMAGE UNDERSTANDING TESTBED

A. Background

The Image Understanding Testbed was established at SRI for the purpose of demonstrating and evaluating research in machine vision sponsored by the Defense Advanced Research Projects Agency (DARPA). Applications of Image Understanding (IU) techniques to automated cartography and military image interpretation tasks have been important components of the DARPA IU research program; these areas form the principal focus of the Testbed project. A number of computer programs developed by participants in the Image Understanding program have been transported to the uniform Testbed environment for examination. These include systems written in UNIX C, MAINSAIL, and FRANZLISP. Capabilities of the computer programs include segmentation, linear feature delineation, shape detection, stereo reconstruction, and rule-based recognition of classes of three-dimensional objects.

B. System Hardware

The principal element of the current Testbed hardware configuration is a DEC VAX-11/780 central processing unit. The Testbed VAX is on the ARPANET network with ARPANET address SRI-IU. The Testbed graphics system centers around a Grinnell GMR275 display system. A DeAnza IP5532 display system is also available at SRI for research and development work. A DEC 2060 central processing unit (SRI-AI on the ARPANET) is accessible from the Testbed VAX via the ARPANET.

The VAX is a four-megabyte system with one tape drive, one RP06 disk drive, three CDC 9766 disk drives, and 32 teletype lines. The VAX interfaces directly to a variety of terminals, a digitizing table, a menu tablet, the Grinnell display system, and a DEC PDP-11/34 minicomputer that controls the DeAnza display system. Other peripherals include a Versatec 11-inch printer/plotter with 200-point/inch resolution, and an Optronics C-4100 color image scanner with resolution selectable from 12.5 to 400 microns. Several different computer network interfaces to the Testbed VAX are in place at this time: an ARPANET link to the SRI IMP, a temporary CHAOSNET interface to an interim Lisp Machine system, and a borrowed 3-megabit/second ETHERNET network to link to other SRI VAX systems. The CHAOSNET and 3-megabit ETHERNET links will be replaced eventually by a 10-megabit ETHERNET interface. This higher-bandwidth link will be used to communicate with second-generation Lisp Machines and other local computer systems.

The Grinnell display system has a resolution of 512 x 512 with 32 bits per pixel. Special features include individual zoom, pan, and color lookup tables on each group of 8 bit planes. The bit planes can be arranged to form a 1024 x 1024 x 8 image of which any 512 x 512 portion can be displayed. The Grinnell system is primarily dedicated to the support of Testbed functions.

The DeAnza refreshed-raster-scan display system also has a resolution of 512 x 512 with 32 bits per pixel. This system is used mainly for research and development. Eight bits each are allocated to red, green, and blue data, and in addition there are eight overlay planes. SRI has installed a special video crossbar system to allow the DeAnza bit planes to be allocated dynamically among our two color monitors and up to eight remote monochrome monitors. All DeAnza graphics operations are carried out by a PDP-11/34 under the direction of the VAX.

C. System Software

The Testbed system runs under the VAX/VMS operating system, using the SRI-developed EUNICE software package to emulate UNIX operating-system services and to support software developed under

UNIX. (UNIX is a trademark of Bell Laboratories.) This combination of operating-system support permits compatibility with both UNIX environments and other VMS/EUNICE environments. In principle all Testbed applications software can be run either on UNIX or VMS/EUNICE systems provided that appropriate system-specific hardware device drivers are available.

The DEC 2060 presently associated with the Testbed VAX runs under the TOPS-20 operating system; this facility is available to run application programs developed on PDP-10 computers that cannot be transported easily to run on the Testbed system itself.

D. Language Support

The high-level programming languages currently used on the Testbed are UNIX C, MAINSAIL, and FRANZLISP. Both DEC and Berkeley UNIX versions of the FORTRAN and PASCAL language compilers are available but are not used in any contributed software. The DEC C language compiler can in principle be used instead of the UNIX C compiler for some applications under the VMS operating system. In addition to FRANZLISP, the ISI VAX INTERLISP dialect and an experimental version of the MIT NIL LISP dialect are also available. Lisp Machine LISP will also be available when Lisp Machines are integrated into the Testbed system.

The Testbed graphics capabilities have been implemented for the DeAnza in MAINSAIL. The Grinnell graphics capabilities are fully supported in C, using software based originally on the CMU Grinnell graphics package. FRANZLISP and MAINSAIL programs may access the Grinnell by means of the C Grinnell graphics package.

E. Personnel

The IU Testbed program has been carried out in the Artificial Intelligence Center of SRI International's Computer Science and Technology Division under the general supervision of Dr. Nils J. Nilsson. Dr. Martin A. Fischler has been the project leader. Dr. Andrew J. Hanson has acted as the IU Testbed project coordinator. Staff members principally concerned with Testbed activities have included D. L. Kashtan and Dr. K. I. Laws. Other members of the SRI research staff who have made essential contributions to the Testbed include Dr. S. T. Barnard, Dr. R. C. Bolles, Dr. A. P. Pentland, Dr. G. B. Smith, and H. C. Wolf. Former members of the SRI vision group who contributed to the IU Testbed effort include Dr. H. G. Barrow, Dr. L. Quam, Dr. J. M. Tenenbaum, and Dr. A. P. Witkin.

II CHARACTERISTICS OF CONTRIBUTED SOFTWARE

A. Overview of Contributions

In addition to SRI International, the institutions contributing software systems to the Image Understanding Testbed are Carnegie-Mellon University, the Massachusetts Institute of Technology, Stanford University, the University of Maryland, the University of Rochester, and the University of Southern California.

The IU Testbed software environment is now reaching maturity. Software modules integrated into the system include libraries of user utilities, graphics routines, and image access routines. Each of the designated Testbed contributor sites has defined and delivered, or arranged for delivery of, contributions to the Testbed system. Among the research contributions are two modules from SRI and two from CMU; also running on the Testbed are one contribution each from Rochester, Maryland, and USC, as well as a major system in FRANZLISP from Stanford. The MIT contributions in Lisp Machine LISP must await the delivery of Lisp Machines to the Testbed. CMU has also furnished utilities, graphics, and picture access packages, while SRI has implemented an extended picture format and many user utilities.

A summary of the currently operational research software contributions is given in Table 1.

Table 1

SUMMARY OF CONTRIBUTIONS

INST.	CONTRIBUTION	LANGUAGE
SRI	Road expert	MAINSAIL
	RANSAC	MAINSAIL
CMU	Picture access and display packages	C
	PHOENIX segmentation system	C
	Stereo/correlation system	C
STANFORD	ACRONYM 3-D model-based vision system	FRANZLISP
MARYLAND	Relaxation package	C
ROCHESTER	Generalized Hough transform system	C
MIT	Stereo reconstruction system	LISP MACHINE LISP
USC	Linear feature analysis	C
	Laws texture analysis	SAIL (C planned)

The following subsections summarize the status of each of the currently integrated contributions.

1. Carnegie-Mellon University Contributions to the Testbed

(1) CMU Grinnell Graphics and Image Packages.

* Date received: 14 August 1981.
* Responsible party: David McKeown.
* Language: C (Berkeley Version 7 UNIX) running on the VAX.
* Description: These packages provide basic access to the functions of the Grinnell display system, as well as the capability of accessing image data files.
* Remarks: A number of minor modifications were needed to make the CMU package work with our specific Grinnell configuration. The present code will support any CMU configuration or the SRI Testbed configuration. The CMU image access package has also been integrated into the testbed environment; a new, extended Testbed picture format has been implemented.

(2) PHOENIX Segmentation Package

* Date received: December 1981.
* Responsible party: Steve Shafer.
* Language: C (Berkeley Version 7 UNIX) running on the VAX.
* Description: This is a segmentation package that uses the Ohlander histogram-partitioning method to segment color imagery. Each pixel in the input image is assigned a segment identification label according to the image characteristics and the parameters selected. Segmentation is carried out hierarchically, with higher-level regions isolated and segmented separately into sub-regions. Segmentation ceases in a given region when the program criteria for significance of the next level of segmentation have not been met.
* Remarks: This system has a sophisticated user interface and a very useful checkpoint system.

(3) Stereo Reconstruction and Correlation Package

* Date received: 28 September 1981.
* Responsible party: Charles Thorpe.
* Language: C (Berkeley Version 7 UNIX) running on the VAX.
* Description: This is a C version of the Moravec correlation and stereo reconstruction package written originally in SAIL at Stanford. The package consists of two portions: CORRELATE, which selects a set of "interesting" points in one image, using the Moravec interest operator, and attempts to locate the corresponding points in a second image, using an efficient hierarchical correlation matcher; STEREO, which uses the same method as CORRELATE to find corresponding points in a series of up to 9 images, and then employs the Moravec method to assign a stereo depth value and confidence level to each member of the set of interest points.
* Remarks: This package implements all the basic capabilities of the original Moravec SAIL system, plus a number of enhancements introduced by Charles Thorpe.

2. University of Maryland contributions to the Testbed

Relaxation Package

* **Date received: Final version received 9 July 1981.**

* Responsible party: Bob Kirby (author: Russell Smith, revised by Joe Pallas).
* Language: C (Berkeley Version 7 UNIX) running on the VAX.
* Description: This relaxation package takes an initial set of probabilities that a pixel belongs to each of a set of classes and iteratively adjusts them according to the class probabilities of neighboring pixels. Two options are provided: a Hummel-Zucker-Rosenfeld relaxation algorithm and a Peleg relaxation algorithm. For the two-class case, the following steps are executed: first, a simple algorithm is used to generate probability assignments from the luminance values in an image; then the relaxation program is used to produce a new assignment of probabilities for each pixel; finally, an inverse algorithm generates a luminance representation corresponding to the reassigned pixel probabilities; the resultant grey scale image can be displayed for the user to monitor the progress of the relaxation process.
* Remarks: A multiclass method of generating probability assignments corresponding to luminance values has been added for test and demonstration purposes.

3. MIT contributions to the Testbed

Marr-Poggio-Grimson Stereo System

* Delivery awaits the installation of the Testbed Lisp Machines.
* Responsible parties: Mike Brady, Eric Grimson and Keith Nishihara.
* Language: Lisp Machine LISP.
* Description: This system uses zero-crossing matches at several scales to compute disparity values between stereo pairs. Interpolation of the three-dimensional surfaces between matched zero-crossings is done with Grimson's interpolation method.
* Remarks: To run efficiently, this system should have special hardware (a convolution box) added on to the Lisp Machine.
* Plans: This system will be integrated with the Testbed when Lisp Machines are incorporated into the environment. If possible a convolution box will be incorporated into the Lisp Machines.

4. University of Rochester Contributions to the Testbed

Hough Transform Package

* Date received: 19 May 1981.
* Responsible parties: Dana Ballard and Bill Lampeter.
* Language: C (Berkeley Version 7 UNIX) running on the VAX.
* Description: This program takes a geometric shape template and attempts to find matching shapes in the image using the generalized Hough transform technique. The matched shapes may differ in displacement, rotation, and scale from the supplied shape template. The most likely values of location, rotation angle, and scale are output by the program and the reoriented templates are displayed over the image.
* Remarks: The CMU graphics package has been used as a basis for incorporating full interactive graphics into this system for both template generation and picture processing. Several improvements have been made in the user interface and in the efficiency of the code, and the package was extended to handle multiple instances of an object.

5. SRI Contributions to the Testbed

(1) Road Expert

* Date received: Approximately 1 Jan 1981.
* Responsible party: Helen Wolf (Lynn Quam available as consultant).
* Language: MAINSAIL running under EUNICE on the VAX.
* Description: This package acquires and tracks linear features such as roads in aerial imagery. Tracking is done automatically in imagery with a known ground truth data base. Once a road is identified and tracked, a separate subsystem is available to analyze road surface anomalies and to place them into categories such as vehicles, road surface markings, and shadows.

(2) RANSAC Image-to-Data-Base Correspondence Package

* Date received: Approximately 1 Jan 1981.
* Responsible parties: Martin Fischler and Bob Bolles.
* Language: MAINSAIL running under EUNICE on the VAX.
* Description: This package selects a best fit to an array of control points possibly containing gross errors. This method offers significant improvements over least-squares fitting techniques if gross errors are present. A typical application is to compute the corresponding camera model, given a set of landmarks in aerial imagery.

6. Stanford University Contributions to the Testbed

ACRONYM System

* Date received: 15 March 1982.
* Responsible parties: Tom Binford and Rod Brooks.
* Language: FRANZLISP running on the VAX. An extensive macro package is used to preserve most of the original MACLISP code.
* Description: ACRONYM takes a scene that has been reduced to a set of two-dimensional ribbons and searches for instances of three-dimensional models that have been supplied to the system as data. This is a rule based system that allows great flexibility in the interpretation and scene-prediction process. Models can also be defined in a very general manner by using generalized cones, constraints, and subclass definitions.
* Remarks: Reduction of an image to a list of ribbons must now be done by hand, starting with a corresponding file of line segments generated by a program such as the Nevatia-Babu line finder. While some test imagery is available with the ribbon reduction already carried out, the Testbed ACRONYM system would profit from the addition of an automated ribbon reduction module.

7. University of Southern California Contributions to the Testbed

(1) Nevatia-Babu Line Finder

* Date received: 1 June 1981 (SAIL version); 14 June 1982 (C version from Hughes Aircraft).
* Responsible party: Ram Nevatia at USC and Julius Bogdanovich at Hughes Aircraft.
* Language: C (Berkeley Version 7 UNIX) running on the VAX.
* Description: This package extracts linear features from an image and produces a data base of lines. The Testbed C version supports 5 x 5 convolution masks configured to identify edges oriented at 30-degree intervals. The edges are then thresholded and linked together into lines.
* Remarks: The C version of this package lacks some of the parallel-line and supersegment extraction features of the SAIL version. This makes the C version less useful for extracting larger linear features with distinguishable edges. It would also be desirable to add support for using a variety of convolution masks. We anticipate that these features will eventually be added to the package.

(2) Laws Texture Analysis

* Date received: 6 July 1981.
* Responsible party: Ken Laws.
* Language: SAIL running under TOPS-20 on the PDP-10.
* Description: This package segments an image by identifying textured regions. It is currently configured as a batch program and uses classification coefficients developed for the particular set of textures in Laws' doctoral thesis.
* Remarks: This program now runs on SRI-AI. Substantial work will be required to make it interact with the Testbed VAX environment. We hope to be able to recode this package in C and to improve its capabilities somewhat.

B. Demonstration, Test, and Evaluation Procedures

Evaluation of software modules on the Testbed has been conducted at several levels, depending on the particular system in question. Each module supports a standard demonstration of its capabilities. The degree to which testing and evaluation can be carried out meaningfully depends on the flexibility of each individual program. Some can run on completely arbitrary images, while others require extensive supporting data that cannot be easily assembled for arbitrary images. Furthermore, some contributions have been extensively documented in existing literature, while others have required additional modifications and documentation regarding their operation in the Testbed environment. This diversity of circumstances has led us to divide the contributions into several groups according to the type and extent of the evaluation and testing to be performed. These categories are described in the following paragraphs:

(1) DESCRIPTION ONLY.

Several contributions cannot be demonstrated interactively on the IU Testbed VAX system, but can be run only on the DEC 2060 or on special hardware such as Lisp Machines. Since such contributions are not part of the integrated Testbed system, their capabilities will be described but not evaluated. The documentation on these modules will cover the following subjects:

* General description of the module and its scientific context.
* Scientific principles of operation of the algorithm.
* References and bibliography.

(2) DEMONSTRATION AND REMARKS.

Several major stand-alone systems need specially tailored data bases to function correctly. When tools for construction of such data bases are not available, the modules will run only on a limited set of images, thus restricting the nature of the evaluation that can be carried out. Such systems will not be systematically evaluated on large numbers of images because of the operational difficulties of setting up the required contexts. These systems will be available for demonstration on limited data sets.

Evaluation issues relevant to these contributions will be treated in documentation covering the following topics:

* General description of the module and its scientific context.
* Scientific principles of operation of the algorithm.
* Program user documentation.
* Suggestions for modifications.
* References and bibliography.

(3) DETAILED EVALUATION.

Those modules that can readily be exercised on a wide variety of imagery in the Testbed image library will be subject to rigorous and detailed investigation. In addition to the information provided for other types of modules, we shall supply a thorough evaluation report on the parameters, performance, strengths, and weaknesses of the module. The detailed evaluation report will include the following:

* General description of the module and its scientific context.
* Scientific principles of operation of the algorithm.
* Program user documentation.
* Evaluation of performance, strengths, and weaknesses.
* Suggestions for modifications.
* References and bibliography.

We present in the following table the level of the evaluation procedure which is currently planned for each of the contributed Testbed software modules.

Table 2

LEVEL OF EVALUATION FOR EACH CONTRIBUTION

CONTRIBUTION	TYPE OF EVALUATION PLANNED
SRI	
Road Expert	DEMONSTRATION AND REMARKS
RANSAC	DEMONSTRATION AND REMARKS
CMU	
Display package	DESCRIPTION ONLY
PHOENIX	DETAILED EVALUATION
Stereo/Correlation	DETAILED EVALUATION
STANFORD	
ACRONYM	DEMONSTRATION AND REMARKS
MARYLAND	
Relaxation	DETAILED EVALUATION
ROCHESTER	
Hough Transform	DETAILED EVALUATION
MIT	
Stereo (Lisp Machine)	DESCRIPTION ONLY
USC	
Linear Features	DESCRIPTION ONLY
Texture Analysis	DESCRIPTION ONLY

C. <u>Summary of Evaluation Results</u>

Major evaluation efforts are in progress at this time on the following modules:

* GHOUGH generalized Hough transform shape-finding system.
* PHOENIX segmentation system.
* STEREO/CORRELATE stereo reconstruction system.
* RELAX pixel-level relaxation system.

In the evaluation process, we have attempted to uncover characteristics of each system which become obvious to the user only through extensive experimentation. While the final results of each of these evaluation studies are not available at this time, we present below some of the salient features of the intermediate results. Full written evaluation reports on each of the above modules will be available soon.

1. GHOUGH

GHOUGH uses the generalized Hough transform method to find instances of a predefined template shape in an image. It allows the determination of the location, scale, and angular rotation of the target object. The system has also been extended somewhat to detect multiple instances of a shape in a single image.

The following templates were used in testing the program: a lake, a right angle, a circle, and an ellipse. Several interesting artifacts of the template parametrization were observed. An example was the quantization of template angles resulting from the use of discrete lattice points to compute the orientation of line segments in the template. Very dense templates generated excessive noise compared to sparser outlines due to the fact that neighboring pixels were related only by angles which were multiples of 45 degrees. This significantly increases the observed noise in the estimated object parameters. Several variations of the implementation strategy have been noted which would reduce such effects.

Other significant characteristics of the algorithm were noted while attempting to locate multiple instances of circular or elliptical storage tanks in a variety of aerial imagery. A profound advantage of the Hough method was found to lie in its ability to discern incomplete and occluded shapes. On the other hand, no single choice of parameters would serve to locate accurately each and every one of the circular tanks obvious to the human observer; the blurred nature of some of the photometry and other characteristics of the tanks (e.g., rounded tops and shadows) required that special choices of parameters and templates be made in order to detect any individual tank. Thus GHOUGH was found to be very useful for detecting unique, photometrically distinguished shapes or partial shapes, but needed higher level information to determine effective parameter choices when less distinctive imagery was available.

2. PHOENIX

PHOENIX is an Ohlander-style segmentation package that uses histogram analysis to carry out a hierarchical segmentation of color imagery. A number of options are available to control the number and type of the segmentation cuts performed on each histogram as well as to select criteria for determining the significance of the segmentation. Noise point merging to smooth out the segments is also supported.

The user interface for PHOENIX is based on the CMU CI command driver, which allows a wide variety of subroutines to be called in an interactive and user-controlled manner. Information about each segment of a processed image can be printed and/or displayed on the graphics system as desired. A variety of switches and flags are available to control graphics and other output from the program. A particularly useful feature is the availability of a checkpoint system that can save the current state of a segmentation process and read it back in at a later date for more detailed examination or additional processing.

A number of fundamental properties of the PHOENIX system have been noted so far in the evaluation process. The best performance is obtained for color imagery in which objects of interest have distinct colors and for which the histograms of one or more spectral bands have at least two distinct peaks. Significant region identification in deeper levels of the hierarchical process also relies on the existence of more than one distinct peak in the histograms of the parent regions. Textured monochrome images often lack these characteristics.

In some instances, reasonable parameter choices fail to produce the intended results; for example, in one test image of a city skyline, the sky itself is segmented into dozens of barely distinguishable subareas, while an obviously colorful American flag is never distinguished from the sky at any level of the hierarchical segmentation process. On the basis of the characteristics of PHOENIX noted in the previous paragraph, one might conjecture that certain types of color imagery that humans can segment well would not be well-suited to PHOENIX. We have confirmed experimentally that histogram-equalized color imagery is easily segmented by humans but not by PHOENIX; the same phenomenon is observed for imagery with significantly different texture regions if the regions have relatively smooth histograms.

PHOENIX can be utilized profitably on imagery which supports color-based region identification if the image digitization has a rich histogram structure. Presumably, certain types of preprocessing could also be performed on some images to adapt them to the PHOENIX domain; in particular, some fairly straightforward transformations of the color space (not supported on the Testbed version of PHOENIX) would be expected to have significant effects. Given appropriate imagery, one could then use the output of PHOENIX for higher-level tasks requiring image segmentation information.

3. STEREO/CORRELATE

The STEREO/CORRELATE package is an image correlation system based on the Moravec interest operator. Moravec's high-performance hierarchical moving-window correlation scheme is used to match interest points in successive images. Conventional stereo reconstruction algorithms are then used to compute a possible stereo depth assignment for sets of matched points.

The correlation algorithm works essentially as follows. A set of points exceeding a given threshold value for the Moravec interest operator is found in one image. The images themselves are all reduced by pixel averaging to lower resolutions such as 1/2, 1/4, 1/8, and 1/16.

The correlation operator then tries to find the best match in each of the other images to a small window around the interest point in the lowest-resolution version of the original image. Searches at the next higher resolution are limited to a window around the best match. Then the process is iterated until a single pixel in each of the full resolution images is identified as the best match.

This process has the obvious advantage of reducing the amount of computation required and improving real-time performance. In typical experiments with the system, one occasionally finds that the search window wanders away from the true matching points and that the search window at the highest resolution level may not contain the desired point. Also, periodic structures such as wall tiles and building windows introduce systematic mismatching of similar interest points. We note that the introduction of some simple higher level guidance could eliminate most such problems with the STEREO/CORRELATE system.

STEREO/CORRELATE functions very well when used to construct a sparse depth map of imagery that contains few confusing or periodic structures; the addition of some external guidance would extend its utility to extremely broad classes of imagery. Among the major advantages of the system are that it has very good real-time performance and behaves well even when applied to poor image data. Ultimately one would like to use the system to construct more detailed stereo depth maps of an image; to accomplish this objective, one would need to add a powerful interpolation system to convert the sparse interest-point map into pixel-by-pixel depth assignments.

4. RELAX

RELAX is a package that supports both the Hummel-Zucker-Rosenfeld and Peleg pixel-level relaxation algorithms. To use the relaxation technique, one first runs a utility that converts a photometric image into a matrix of probabilities that given pixels belong to postulated categories. A set of compatibility coefficients is then computed to support the relaxation computations. Finally, one performs any desired number of iterations to yield a new set of probability assignments; if desired, a revised grey-scale image can be computed from the probability assignments and displayed.

The various steps in the application of the RELAX package have been integrated into a flexible user system based on the SRI ICP interactive command processor. We note that two-category relaxations allow fairly straightforward conversion between the imagery and probability structures; three or more categories require methods that are increasingly arbitrary. One solution to this problem would be an interactive utility to aid the user in assigning category probabilities.

This system's ability to improve noisy imagery and to aid in the extraction of an image signal depends strongly on the nature of the initial data and the probability assignments. The current system works most robustly on images with a two-category interpretation, and hence is best used on image segments with a single bright signal area against a dark background (or vice versa). The most effective way to use this system would be first to identify a subarea containing only one object of interest against a bland background, then to run RELAX to improve the signal. Alternatively, one could use an application-dependent preprocessor to assign probabilities based on criteria more complex than the values of single pixels. This system produces excellent results if sufficient information is available for a meaningful assignment of category probabilities to the pixels of the original image.

III TRANSPORTABLE FEATURES OF THE TESTBED ENVIRONMENT

One of the objectives of the Testbed program has been to lay the foundations for a system that would be in some sense transportable to other similar research environments. This transportability would allow other sites to make use of existing Testbed code without having to develop their own version; it would also make it possible for other sites to carry out their own evaluations and improvements of basic Testbed contributions to meet their specific needs.

These objectives have been partially met. Each contribution to the Testbed system can be tested and demonstrated with minimal modifications on UNIX or EUNICE/VMS VAX systems with Grinnell display devices. Many utilities have been acquired from contributing sites or developed locally by Testbed personnel. A picture library and a simple system for accessing it have been developed. A new and general Testbed image file format has been created that supports all of the image types we have found useful in integrating contributed software. A modified version of the CMU image access package supports all essential image retrieval and access functions.

There are also several desirable objectives that remain to be achieved at this time. For example, graphics and image display on the Testbed are supported entirely by an extension of the CMU Grinnell display package. This is a large body of software whose existence allowed basic Testbed demonstration and testing objectives to be met in a timely fashion. However, the package is manifestly device-dependent, and so each of the application programs carries with it the device dependence inherent in using the Grinnell display package.

Ultimately, we would like to adopt a uniform device-independent graphics standard to support the Testbed demonstrations on whatever devices happen to be available at a particular site. This goal has been hampered by the well-known deficiencies of the SIGGRAPH standard in the area of raster graphics as well as the unsettled status of proposed extensions and substitutions for the SIGGRAPH format; in particular, the ANSI standards committee may soon decide to endorse an entirely different standard. A device-independent system will soon be implemented on the Testbed for use in the SRI research program, but the exact choice of format is still being debated.

Another objective is the establishment of a standard set of utilities for registering multiple images to a ground truth data base. Some progress has been made in this direction by the SRI RANSAC system and some supporting modules which are currently being converted into C, and by the CMU "Browse" system (which is not yet ready for transport to the Testbed). Further systematization of such image generation data as time of day, lighting characteristics, photometric parameters and camera characteristics would also be desirable. The systematic application of IU techniques to cartographic problems requires that such information be available for all imagery intended for use as source data.

In the following subsections, we present a summary of the basic capabilities which are supported in the Testbed system and are potentially transportable to copies of the Testbed system.

A. Utility Programs

Among the generally useful utility programs available on the Testbed are the following:

(1) CI. This is a command interpreter contributed by CMU. It allows one to link a variety of subroutines into a top-level command processor and to invoke the subroutines with arguments provided interactively by the user. Extensive help and utility facilities are supported.

(2) ICP. This is a command interpreter for the C language contributed by SRI. It is very similar to CI, except that its treatment of arguments and local variables is more general. ICP, for example, is able to invoke system or user subroutines directly, while CI must have an argument-parsing interface written for each routine.

(3) DOC. This is a CMU utility for generating program documentation (UNIX "man" entries) without needing to know any details of the TROFF phototypesetting system. All information that the program needs to generate a syntactically correct "man" entry is obtained by interrogating the user.

(4) NORMALIZE. This CMU routine normalizes a grey-scale image to produce a new output image with desired compression or clipping. SRI modifications allow grey-scale stretching as well.

(5) REDUCE. This CMU routine extracts a subwindow of an image, or rescales an image by an integer sampling factor.

(6) SHAPEUP. The original CMU routine of this name has been entirely rewritten to support conversions among many image formats.

(7) IMGSYS. This is a Testbed system that allows a variety of picture files to be accessed, described, and displayed in a desired format of subwindows on the display system.

B. User Interface Systems

The following systems permit useful information or features to be made available to users of the Testbed:

(1) TESTBED DEMO DIRECTORIES. A collection of complete demonstration facilities have been set up in the Testbed demonstration directory. Each of the demonstratable C-coded contributions is represented by a series of subdirectories supporting various informative demonstrations of the program capabilities. Ground truth data for comparison with the program output is also available in some cases. The command files supplied in the demonstration directories provide detailed examples of program invocation; from these examples, a sophisticated user can in principle deduce the fundamental operating procedures for each program. We note that *detailed written* documentation of program usage will be available in the evaluation reports for each contribution.

(2) VAX EMACS INFO. An INFO macro package has been developed at the SRI Testbed to support an extended version of the TECO EMACS INFO system. This system is a chain-linked documentation reading and generation system that utilizes the basic window-oriented features of the EMACS editor to access, search, and display text information. On-line Testbed documentation is available via the INFO system. This provides a well-structured and convenient access mechanism to the on-line documentation of the system's functions and capabilities.

(3) LEDIT and LTAGS. An intercommunicating pair of special modifications of EMACS and FRANZLISP have been implemented on the SRI EUNICE/VMS system to support Lisp-Machine-like capabilities for developing FRANZLISP programs. LEDIT allows the user to copy any defined function from a FRANZLISP image into an EMACS editor buffer, modify it, and reload it into the FRANZLISP process without changing any other part of the

FRANZLISP environment. Files with many functions can be edited in EMACS and the functions of interest marked for loading when the user returns to FRANZLISP. The LTAGS package works in concert with LEDIT in EMACS, allowing the user to display any desired function in his window for editing by simply giving the first few characters of the function name; the system automatically keeps track of which files contain which functions, so that such complications are invisible to the user. We note that the system service capabilities needed to support the intercommunications involved in LEDIT are not now available on UNIX, and so require the VMS operating system.

(4) ASKLIB and ARGLIB. This is a set of utility routines available for application programs that need to interrogate the user about the values of program parameters. ASKLIB has been slightly extended from the original CMU package, while ARGLIB has been entirely rewritten for the Testbed.

(5) ERR.H error package. This is a Testbed package that supports flexible and user-friendly reporting and handling of error conditions.

C. Picture Data Base System

The Testbed Picture Data Base System (PICDBMS) is a FRANZLISP-based system that interacts with a directory of test imagery to allow the entry and retrieval of image characteristics from an image data file. Following the CMU picture file conventions, each image is assigned a named directory, e.g., /iu/tb/pic/chair, which contains the picture data, e.g., 4red.img, 4blue.img, 4green.img., along with collateral data files. PICDBMS contains utilities for creating or editing a "pic.dat" file in each picture directory. This data file, containing data formatted for easy LISP readability, includes picture descriptions, picture characteristics, and a list of data base keys. Typical data base keys that are currently supported include the words listed in parentheses below:

* IMAGE TYPE AND MULTIPLICITY: (bw color stereo multiple)
* SCENE DOMAIN TYPE: (indoor cultural natural)
* CONTENT CHARACTERISTICS: (point linear area)
* VIEWPOINT: (aerial ground).

Other types of data can be supported as the need arises. Sets of images can be retrieved by asking for images corresponding to a set of keys; both AND and OR conditions are supported in the data base key interrogation.

Additional facilities of PICDBMS include a browsing utility to display lists of images provided by the keyed data base retrieval subsystem. Images too large to fit on one Grinnell screen can have any desired windows displayed in sequence.

IV PLANS

The future of the Image Understanding Testbed program at SRI will be closely tied to the SRI IU research efforts and to the evolving characteristics of Testbed copy systems to be installed at ETL and potentially at other DMA sites. The generality and transportability of the IU programs and utilities will continue to be enhanced in support of various research efforts. We anticipate that the incorporation of Lisp Machines into the environment will result in a substantial movement toward LISP-based IU application programs.

The major shift in emphasis in the Testbed environment at SRI will be from low-level image processing code towards increasing reliance on rule-based expert systems to control the choice of low-level processes, the parameters to be used, and the communications interfaces between the computer system and the human analyst. We anticipate development of a substantial capability for supporting expert systems that facilitate the application of IU research results to cartographic problems.